U.PORTO

MESTRADO
MULTIMÉDIA - ESPECIALIZAÇÃO EM TECNOLOGIAS

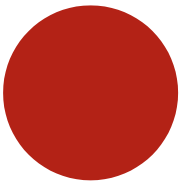
A framework for the manipulation of video game elements using the player's biometric data

Manuel César Bessa Seixas

M
2016

FACULDADES PARTICIPANTES:

FACULDADE DE ENGENHARIA
FACULDADE DE BELAS ARTES
FACULDADE DE CIÊNCIAS
FACULDADE DE ECONOMIA
FACULDADE DE LETRAS



FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

A framework for the manipulation of video game elements using the player's biometric data

Manuel César Bessa Seixas

Mestrado em Multimédia

Orientador: Pedro Cardoso (Assistente Convidado)

Co-orientadores: Miguel Carvalhais (Professor Auxiliar)

Rui Rodrigues (Professor Auxiliar)

Julho de 2016

A framework for the manipulation of video game elements using the player's biometric data

Manuel César Bessa Seixas

Mestrado em Multimédia da Universidade do Porto

Aprovado em provas públicas pelo Júri:

Presidente: Nuno Honório Rodrigues Flores (Professor Auxiliar)

Vogal Externo: Carlos José Ribeiro Campos (Equiparado a Assistente 1º Triénio)

Orientador: Pedro Jorge Couto Cardoso (Assistente Convidado)

---

15 de Julho de 2016

Abstract

The following dissertation focuses on the use of several physiological signals produced by human players to manipulate several game elements. These elements comprehend the game mechanics, aesthetics, behaviour of elements that act in opposition to the player, dialogues and narrative, among others.

Our objective is the development of a framework aimed at supporting games that use, or intend to use, biofeedback. Looking for a solution (and inspiration), we studied how biofeedback is being presently applied, either in scientific experiences or commercially, type of biometric sensors are available in the market, other frameworks with similar purposes, as well as *profiling* and dramatic tension in games.

The result is an application able to collect the players' biometric data, as well as their affective state, turning it into values usable by the game; and a rules system with the aim of streamlining this data handling.

Keywords: biofeedback, affective feedback, affective gaming, framework, video games

Resumo

A presente dissertação foca-se na utilização dos vários sinais biométricos produzidos pelo ser humano (neste caso específico, os do jogador) para manipulação dos vários elementos existentes num videojogo. Estes elementos abrangem as mecânicas de jogo, estética, comportamento dos elementos que actuam em oposição ao jogador, diálogos e narrativa, entre outros.

O nosso objectivo é o desenvolvimento de uma *framework* direccionada ao suporte de jogos que pretendam utilizar, ou utilizem, *biofeedack*. À procura de uma solução (e inspiração), estudou-se a forma como o *biofeedback* é empregue na actualidade, na área científica e comercial, os tipos de sensores biométricos disponíveis no mercado, outras *frameworks* com objectivos semelhantes, bem como *profiling* e tensão dramática em jogos.

O resultado é uma aplicação capaz de captar a diversa informação biométrica proveniente do jogador, bem como o seu estado afectivo, transformando-os em valores utilizáveis pelo jogo; e um sistema de regras cujo o intuito é facilitar e agilizar a utilização destes dados.

Palavras-chave: *biofeedback*, *feedback* afectivo, jogos afectivos, *framework*, videojogos

Agradecimentos

Gostava de agradecer ao meus orientadores Professor Rui Rodrigues, Professor Miguel Carvalhais e Professor Pedro Cardoso por toda a ajuda que me deram ao longo desta dissertação. Um agradecimento especial ao Professor Pedro Cardoso (e um pedido de desculpas) por todos os erros ortográficos (não eram de propósito, eu juro!).

Manuel César

Contents

List of Figures

List of Tables

Acronyms and Symbols

2D	Two dimensions (two-dimensional space)
A.I.	Artificial Intelligence
API	Application Programming Interface
C#	C Sharp (a programming language)
ECG	Electrocardiography
EMG	Electromyography
GSR	Galvanic Skin Response
GUI	Graphical User Interface
HCI	Human-Computer Interaction
HD	High definition
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
NPC	Non-playable character
OS	Operative System
OSC	Open Sound Control Protocol
OSes	Operative Systems (plural of OS)
SCL	Skin conductance level
SDK	Software Development Kit
SFX	Special effect(s)
UI	User Interface
XML	eXtensible Markup Language

Chapter 1

Introduction

Video games have been evolving since the moment they were born. As time goes by new technologies appear, allowing for an improved or more diverse gaming experience, eye-pleasing graphics, sophisticated artificial intelligence (A.I.), more complex and interactive storytelling methods, pleasing sounds and realistic soundtracks, virtual reality, among so many other improvements.

One of those many improvements was biofeedback, a technology that uses the players' physiological information (pulse, secretion of sweat, body temperature are examples of these) with the purpose of altering its experience while playing as well as allow a set of new inputs that can be used by the players to control the system.

However, biofeedback is no novelty: industry is doing their best to create new input devices that are able to take advantage of this new interaction process; and scientific experiments (either in the medical, in human-computer interaction or in video game fields) using biofeedback already thrive. Unfortunately, it is yet not possible to say if this is the future standard of the gaming industry or simply a momentary trend, that will fade with the passage of time. Nevertheless, it is a step in a new direction!

This work is inserted in the fields of biofeedback, affective feedback[1] (or affective gaming). Biofeedback in games consists on using the players' physiological values to control the game system; and affective feedback consists on using the players' emotional state to influence the game, both core aspects of our framework.

It could also be argued that it also belongs to the field of human-computer interaction, since affective feedback and biofeedback portray an interaction between the machine and the

[1] Since affective feedback is sub-field of affective computing, we could also argue that our work also belongs to the main field of study.

human. However, our focus lies on the process and not on the interaction itself, leading us to believe that this work outside of this field scope.

1.1 The initial idea

Initially, our objective was combining biofeedback with video game narrative. Until now, biofeedback had been applied to some games but, narrative-wise, there's yet to be a study evaluating the benefits/improvements (or lack thereof) of this type of input in the narrative of a game. Our objective was to unite both and create a new and appealing experience for the player.

A first step would be to look for tools or software solutions that would allow us to develop this idea. However, we ran into a predicament. Experiments performed until now were developed from core and focused on specific sensors and were not bothered with the affective component; or abstracted the capture and parsing of the sensor data by using expensive technology, dealing only with the implementation of the game and its mechanics; or described a set of rules and principles for creating an affective feedback solution, with the authors using these to create a prototype but without providing any of the tools used in the process. In other words, we were unable to find a solution that would cover the development and implementation workflow, making the initial objective unfeasible, forcing us to abandon this idea.

1.2 Relevance and study's contribution

While these obstacles made us renounce the original idea, they revealed a important shortcoming in the area: the inexistence of a tool or solution able to abstract the whole setting-up process. A tool such as this one would allow game designers and game developers to focus their attention exclusively on the game, saving resources and speeding development.

This solution takes the form of a framework, that fulfils the following conditions:

1. Is able to receive and parse information (the player physiological data) provided by the sensors. There must not be a restriction in number or type of sensors;
2. Uses the collected data to determine the players' current emotional state;
3. Makes this information available to the game (sensors data and emotional state) so it can be directly used by the game developers and designers;
4. Offers a semi-automatic way of converting this information into values or actions understandable by the game.

As such, we could say that our study's contribution would be an abstraction of the setting-up process. Perhaps not all, but a significant part of it.

1.3 Objectives of the work

Based on this, we could say that our core objectives for this work consists on:

1. Conceptualizing a framework that allows the manipulation of video game elements using the players' biometric data (including their emotional state) and its implementation.
2. Present a proof of concept: a small example (a game prototype) that employs the developed framework, as a way to test, debug and validate it.
3. Perform an experiment, targeted at game designers and developers, to determine if the framework is indeed a useful asset and collect feedback for improvements and new functionalities.

1.4 Methodologies

Regarding our research, we first focused on finding meanings of *emotion* and how to represent it. Emotion is a word commonly used in a day-to-day basis for describing diverse phenomena which, when transported to the scientific field, are considered irreconcilable. We also focused our attention in the fields of affective gaming, affective feedback and affective computing, hoping to understand their distinction, as these could turn out to be powerful keywords when researching.[2]

With biofeedback being a core aspect of this work, our investigation also included an extensive search about what type of biometric sensors were available in the market. At some point in our work, we would require one or more sensors to test our framework and develop a game prototype.

Finally, for the development of our framework, we focused on which tools or existing frameworks could help developing ours and game mechanics. Game mechanics had a triple advantage: 1) they gave us an insight on how biofeedback is being used on games, 2) provided clues to some functionalities our framework should have and 3) were clear examples of game mechanics we could implement on the game prototype.

1.5 Dissertation structure

This document is divided in six chapters. The first chapter is the introduction chapter, introducing the relevance, objectives and methodology employed in this study.

The second chapter focuses on the state of the art on the research done for the development of this project, including the definition of emotion; a distinction between biofeedback, affective

[2] In the end, these ended up having a similar meaning.

feedback and affective computing; and physiological signals that could be captured from the player. This information is complemented by frameworks with a similar objective as ours, profiling and dramatic tension in game.

The third chapter details the concept behind the framework that was developed: its components, how they work and possible variations. This chapter is also complemented by the a set of video game mechanics used in games with biofeedback, with suggestions of new[3] ones from our part.

The fourth chapter describes the implementation of our framework or, in other words, its transformation from a concept to *middleware*. The chapter details which technologies were used, graphical user interfaces developed and how each component was given form.

The fifth chapter presents a case study: a simple example where our framework was applied to an existing game, allowing us to test how complex is the setting up process, find errors or other unexpected behaviour and possible improvements for other iterations of the framework.

And in the sixth and final chapter we present the conclusions of all our work: which objectives were achieved, limitations and improvements to perform in the framework (either for the concept or software).

[3] They are, in fact, adjustments made to game mechanics so that they are able to support biofeedback.

Chapter 2

State of the Art

In this chapter will focus on describing what is *emotion*, its definition and how to represent it, as well as the scientific areas of *affective computing* and *affective gaming*, distinguishing both and mentioning important concepts of each, as these are the pillars of *affective* feedback and biofeedback.

As a core theme of this project, the area of *biofeedback* will be given particular attention, presenting the formal definition of this term, detailing the different types of signals yielded by the human body,[4] and ways of capturing data based on those.

Finally, we will also explore the subject of *profiling*, explaining what can be achieved through it, as a way to give meaning to the biosignals collected from the player, and *dramatic tension*, a theme that at first may seem to be outside the scope of this project and yet contains a rich pool of ideas from which to extract game mechanics to use in our work.

2.1 Finding emotion

When developing this work, one of the main difficulties was finding a scientifically correct definition for the term *emotion*, as it is used in the day-to-day basis. This section will try to differentiate the several terms as we try to discover a possible solution for this problem.

This conflict is no novelty as Daniel Batson, Laura Shaw and Kathryn Oleson already mentioned that the terms *affect*, *mood* and *emotion* were used interchangeably (1992 qtd. in Ekkekakis 2012, 321) at the time their work was published. However, Panteleimon Ekkekakis claims that a differentation is starting to appear (2012, 322).

[4] Also know as physiological data. For the sake of simplicity, from now on, we will call these signals: *biosignals*.

Simon Droog also highlights the previous facts, stating that the word *emotion* is often applied to distintic phenomena such as *passions*, *sentiments*, *temperament* and *moods* although, he claims, these could be grouped together inside the phenomema called *affective states* (or *emotional states*). *Affective states* can be divided in: *emotions*, *moods*, *emotional traits* and *sentiments* (What is emotion? (part 1) – 4 Affective states 2010).

2.1.1 Core Affect

Core affect is defined by James Russell and Feldman Barret as a" neurophysiological state consciously accessible as a simple primitive non-reflective feeling most evident in mood and emotion but always available to consciousness" (2009 qtd. in Ekkekakis 2012, 322). *Pleasure*, *displeasure*, *tension*, *relaxation*, *energy* and *tiredness* are examples of core affects (Ekkekakis 2012, 322).

Ekkekakis also claims that a person is always experiencing *core affects,* although they vary in intension and nature over time. They can be part of an *emotion* or *mood* although they can also occur isolated (2012, 322).

2.1.2 Emotion

When looking for a definition for the word *emotion*, The Free Dictionary[5] provides us with three definitions:

- ″A mental state that arises spontaneously rather than through conscious effort and is often accompanied by physiological changes; a feeling: the emotions of joy, sorrow and anger."
- "Such mental states or the qualities that are associated with them, especially in contrast to reason: a decision based on emotion rather than logic."
- "A complex and usually strong subjective response, such as love or hate..."

From an academic point of view, the definition of *emotion* varies greatly. According to Kai Kuikkaniemi et al., emotions are "biologically based action dispositions that have an important role in the determination of behavior" and are composed by: *subjective experience*,[6] *expressive behaviour*[7] and *physiological activation*[8] (2010, 859). Mike Ambinder describes it as a

[5] The Free dictionary, by Farlex, is a free online dictionary that can be found at: http://www.thefreedictionary.com (accessed February 8, 2016).

[6] Named *subjective feeling* by Droog, *subjective experience* is "the conscious awareness of the emotional state one is in," with each emotion involving "a specific feeling which is a basic, irreducible kind of mental element" (What is emotion? (part 3) – 4 Ways of Manifestation 2010). Examples are: feeling happy, feeling inspired or feeling joyous.

[7] *Expressive behaviour*, named *expressive reaction* by Droog, are the changes provoked by an emotion in the face, voice and posture of an individual, with each having a particular pattern: "For example, anger comes with a fixed stare, contracted eyebrows, compressed lips, vigorous and brisk movements and, usually, a raised voice, almost shouting" (What is emotion? (part 3) – 4 Ways of Manifestation 2010).

"subjective, internal state induced by response to (usually) external events" that can be represented as a vector, whose magnitude represents arousal and whose direction represents valence (2011).

For Russell and Barret, *emotion* is "a set of interrelated sub-events concerned with a specific object" (1999 qtd. in Ekkekakis 2012, 322). According to Ekkekakis , this object could be a person, event or a thing, real or imaginary, from past present or future. Some of the common components of an *emotion* are: 1) a *core affect*, 2) a behaviour matching the emotion, 3) focus on the eliciting stimulus, 4) an analysis of the meaning and implications of the stimulus, 5) the assignment of the consequence (emotion) to the cause (stimuli), 6) the experiencing of the emotion and 7) the neural and endocrine changes consistent with it (2012, 322).

For Droog, *emotions* possess the following characteristics: are *intentional*, the *subject is easily identifiable*, are *acute*, *last a short period of time* and *have a cause*. They are *intentional* because they involve a relation between the one who experiences them and the subject who triggered them,[9] with this subject being easily identifiable.[10] Regarding their lasting period, emotions can persist from seconds to minutes at most, no longer. The cause is a stimuli[11] which the one feeling the emotion is sometimes unable to identify (What is emotion? (part 1) – 4 Affective states 2010).

Robert Plutchik presents his definition of emotion based on the evolutionary theory, by Charles Darwin. From this point of view, emotions are like a mechanism created to allow the animal (in this case, humans) to survive, increasing the *evolutionary fitness*.

> An emotion is not a simply a feeling state. Emotion is a complex chain of loosely connected events that begin with a stimulus and includes feelings, psychological changes, impulses to action and specific, goal-directed behavior. That is to say, feelings do not happen in isolation. (...) Extending Darwin's idea a bit, I propose that in general, emotions are activated in an individual when issues of survival are raised in fact or by implication. Such situations include threats, attacks, poisonous substances or the sighting of a potential mate. The effect of the emotional state is to create an interaction between the individual and the event or stimulus that precipitated the emotion. The interaction usually takes the form of an attempt to reduce the disequilibrium and reestablish a state of comparative rest. (...) Emotions are not simply linear events, but rather are feedback processes. (Plutchik 2001, 345 - 347)

---

[8] Also named *physiological reaction* by Droog, is " is the change in activity in the autonomic nervous system which accompanies emotions" (What is emotion? (part 3) – 4 Ways of Manifestation 2010). Examples of this are pupil dilation, sweat production and a increase in heart rate: all automatic actions governed by the autonomic nervous system.

[9] This subject could be an event, object or surrounding.

[10] As Droog claims: "one is afraid of something, proud of something, in love with something and so on" (2010).

[11] This stimuli can take the form of an event happening in the surroundings ("someone calling for us") or a change within us (resulting from a thoughts or memories) (Droog, What is emotion? (part 1) – 4 Affective states 2010).

The author also complements this definition, by stating that the origin or trigger of *emotion(s)* is a stimulus, either *external* (e.g. a predator is close by) or *internal* (e.g. dreams), that place the individual in a state of disequilibrium. To return the individual to the previous state (a state of equilibrium) an impulse to action is generated in order to contradict the stimuli or event: "[f]or example, running from a source of threat reduces the threat and tends to reestablish the condition that existed before the threat" (2001, 348). These impulses to action might take some time to manifest, or not manifest at all, due to embarrassment or fear of retaliation. (2001, 348).

Although there are many definitions for *emotion* there is yet to be a single and globally accepted one. Several authors like Nelson Zagalo and Plutchik reiterate this. Plutchik claims that "more than 90 definitions of 'emotion' were proposed over the course of the 20th century. If there is little consensus on the meaning of the term, it is no wonder that there is much disagreement among contemporary theoreticians concerning the best way to conceptualize emotion and interpret its role in life" (2001, 348). Zagalo presents two reasons to justify this dilemma, quoting Sherry Turkle in the process: "a kind of collective fear that protects the last frontier of what separates us [humans] from machine and other animals"[12] (2009, 31) and as a complex term, used in many distinct fields of knowledge, each more worried with defining the concept so it matches their methodological scope than to find a concept that encompasses them all.

2.1.3 Mood

Moods, according to Ekkekakis (2012) and Droog(What is emotion? (part 1) – 4 Affective states 2010), are largely different than emotions, although both are acute (they arise suddenly and manifest intensely) and have a cause. In terms of differences, *moods* tend to last longer than emotions, having a relatively long-term character although they are limited in time;[13] are *non-intentional*[14] and are *not directed at a particular subject*.

> For example, when a person is in an anxious mood, the object might be something as general as the whole future or as distant as life in 20 years; when a person is in a depressive mood, the object might be the totality of self; and when a-person is in an irritable mood, the object could be anything and anyone. (Ekkekakis 2012)

[12] From the original: "*uma espécie de receio colectivo defensor da última fronteira que nos separa da máquina (Turkle, 2004) e em certa medida dos outros animais.*"
[13] "One can be sad or cheerful for several hours or even for several days" (Droog, What is emotion? (part 1) – 4 Affective states 2010).
[14] As previously shown, Droog identified that emotions are intentional because they involve a relation between the one who experiences them and the subject that triggered them, with this subject being easily identifiable.

Moods, like emotions, are caused by a trigger. However, these triggers are often the *result of the combination of a number of causes* making it difficult for the individual to specify the cause that originated the particular mood (Droog, What is emotion? (part 1) – 4 Affective states 2010). Other times, the eliciting stimuli *could be temporarily distant*[15] (Ekkekakis 2012).

A *mood* can also be inconspicuous, since an individual is sometimes unaware of being in a certain mood.

2.1.4 Emotional Traits

According to Droogs (What is emotion? (part 1) – 4 Affective states 2010), *emotional traits* can be seen as characteristic mood of an individual ("one can have a cheerful or a gloomy character"). These also tend to have a longer lasting period. Like moods, *emotional traits* are non-intentional.

2.1.5 Sentiments

Sentiments are the likes and dislikes of an individual, his/her attitude toward another individual, object or event. Similarly to emotional traits, they also have a long lasting characters, which may persist through a life time, and they also involve a relation between the feeler and the subject, akin to emotion (Droog, What is emotion? (part 1) – 4 Affective states 2010).

One important point highlighted by Droog is that *sentiments* tend to be confused with emotions: "[s]ome examples of sentiments are 'I am afraid of dogs' or 'I love ice-cream'. (...) being afraid of dogs (sentiment) and being frightened by a dog (emotion), are essentially different states."

As stated, one core term of our research is *emotion* and, as such, it was important to give it a clear and precise definition. Yet, this was not possible, as the word is used in very different contexts with very different meanings. To tackle this problem we could: 1) create a new definition for the term, which would only add more confusion to the already unclear term, thus increasing the problem; or 2) use a definition proposed by one of the mentioned authors.

Initially we choose Ambinder's definition as it provided a hint on how to translate this information to a digital medium, something important to this work as emotions will have to be collected, represented and stored in a format acceptable by a computer, which is unable to directly process diffuse and continuous information. However, at a later time, we went back to

[15] "(...) a person can wake up in a bad mood in the morning as a result of a confrontation the previous evening" (Ekkekakis 2012, 322).

this question, and choose Droog's *affective states* (or *emotional states*), as his concept encompasses several others and is able to emulate the vagueness with which the word *emotion* is used. Nevertheless, we still kept Ambinder's computational representation in mind as it would prove useful for the implementation of our framework.

Finally, it is important to remind the reader that this choice is made in order to simplify the problem at hand and not with the intent to declare this as the best, correct or definite solution.

2.1.6 Two types of emotion

Zagalo argues the existence of two types of emotions: the *basic* (also called *primary*) and the *secondary*, although there still are discussions in regards to this differentiation.

For him, basic emotions appear to be universal, are born with us and take place almost automatically and unconsciously(2009, 47-48), with these being *happiness*, *sadness*, *fear*, *anger* and *disgust*. As a base for his claims, the author references António Damásio and Paul Ekman perspectives, with whom Oatley and Johnson-Laird also agree: "[t]his way, all agree with the differentiation between basic and secondary emotions as they agree with the nomination of the same five emotions: *Happiness, Sadness, Fear, Anger and Disgust*"[16] (2009, 47). Zagalo adds that, in 1984, Ekman identified *Surprise* as a basic emotion only to back down on his actions, in 1999.

Secondary emotions, on the other hand, are a mix of basic emotions, are learnt and are the result of the cultural environment the individual is exposed to. These emotions also depend on the moral judgement of the person, do not occur as automatically as the primary and possess a more lasting character (Ekman 1999 qtd. in Zagalo 2009, 48). Examples of these are g*uilt*, *shame* and *envy*.

A summary of differences between these two emotions can be found on **Table 1**.

Table 1: Primary and secondary emotions based on Zagalo's definition.

	Primary emotions	Secondary emotions
Are universal?	Yes	No
Acquirable?	No. Are born with us.	Yes. Are learnt.
Trigger?	Automatic and unconscious.	Less automatically than primary. Depend on the cultural environment

[16] From the original: "Desse modo, todos concordam com a diferenciação entre emoções básicas e secundárias assim como concordam na nomeação das mesmas cinco emoções: Alegria, Tristeza, Medo, Raiva e Nojo."

		and individual moral judgment.
Lasting?	Last less than secondary.	Yes
Examples	Happiness, Sadness, Anger, Fear, Disgust	Guilt, Shame, Envy

However, one would expect a concise list of primary and secondary emotions, but that does not happen. Plutchik in his emotional model,[17] opts to use strikingly different emotions[18] from the ones presented by Damásio, Ekman and Zagalo. Just like in emotion, we can see that the classification of primary and secondary emotions is still unclear.

2.2 Representing emotion

In the course of this work, three models for representing affective states were found: Russell's, Plutchik's and Lövheim's, which are described and presented bellow in the same order.

2.2.1 Russell's circumplex model of affect

One way of representing affective states is using Russell's *circumplex model of affect*, created by James A. Russell, in his article in 1980, as can be seen in **Figure 1.** The author argues that the affective states appear from the combination of two *neurophysiological systems*, namely *valence* and *arousal* (1980). In other words, "[e]ach emotion can be understood as a linear combination of these two dimensions, or as varying degrees of both valence and arousal" (Posner, Russell and Peterson 2005, 715).

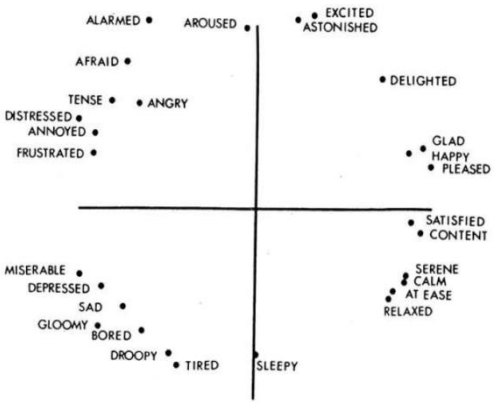


Figure 1: Russell's initial circumplex model of affect (J. A. Russell 1980)

[17] Plutchik's model of affect will be explained in section **2.2.2**.
[18] These emotions can be seen in **Figure 4**.

As stated, the previous model is able to represent affective states based on *valence* and *arousal*, represented in a two-dimensional referential, with the horizontal axis representing the *valence* and vertical axis representing the *arousal*. *Valence* expresses if the feeling is pleasant or unpleasant, while *arousal* can be seen as the level of alertness of the person or the quantity of movement/energy the affective state demands.

As seen on **Figure 1**, *excited* results from a mixture of high arousal (lots of movement and agitation) and a positive (pleasant) valence. *Sleepy*, on the other hand, presents a very low arousal (not too much action or energy is demanded) and a slightly positive valence. Contrastingly, *miserable* is an affective state with a slightly low arousal and a very negative (unpleasant) valence.

In 2005, Posner, Russell and Peterson propose a new design for the circumplex model of affect, as can be seen in **Figure 2**. Differences appear to be purely aesthetic, with the arousal axis now being called activation and deactivation. The affective states presented on the circle are also similar to the original although some have been removed,[19] others renamed[20] and others relocated to a new position.[21]

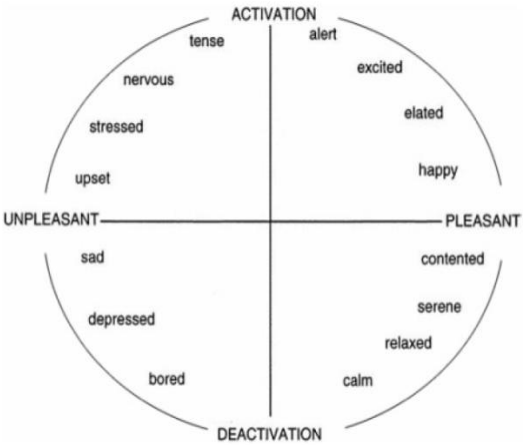


Figure 2: The circumplex model of affect (Posner, Russell and Peterson 2005)

These authors also make an interesting remark: "although adults are fully able to differentiate affective states in both the valence and arousal dimensions, the circumplex [model of affect] in children is poorly differentiated, with affective states grouped primarily according to extremes of valence and with little discrimination on the arousal dimension" (2005, 729). This should be considered when applying this model in contexts involving children.

Finally, according to Zagalo, this model can be simplified, dividing the circumplex model of affect in four quadrants, as can be seen in **Figure 3**. These represented a more generic

[19] *Astonished* is one example.
[20] Assuming that, *content* as been renamed *contented*, for example.
[21] *Tense* had its arousal increased and its valence decreased when compared to the original, which had a lower arousal and a more negative valence.

affective state, with the top-left quadrant identifying *tension*, the top-right *happiness*, the bottom-left *sadness* and the bottom-right *calm* (2009, 52).

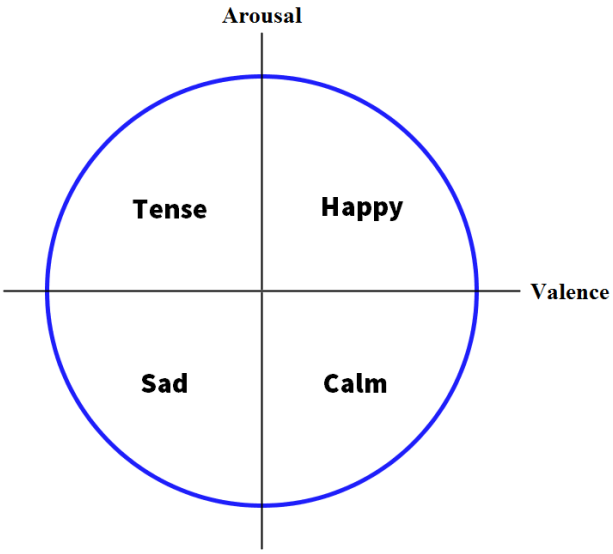


Figure 3: Zagalo's simplified circumplex model of affect.

2.2.2 Plutchik's circumplex model of affect

Plutchik (2001) also presented his own circumplex model, matching his concept of emotion. Contrary to Russell's, this is three dimensional and is represented as cone. It can, however, be exploded to achieve a two dimensional model, as seen in **Figure 4**.[22]

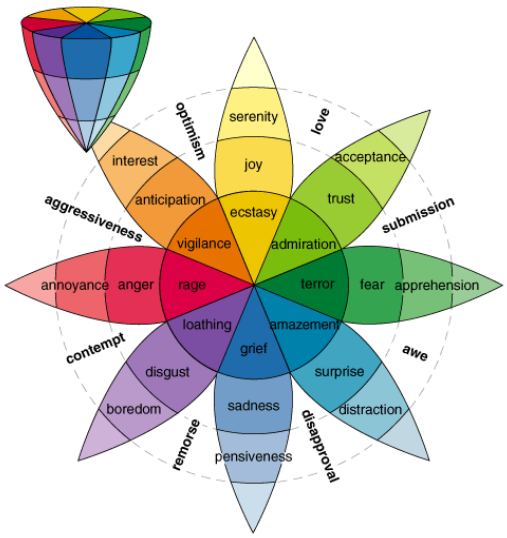


Figure 4: Plutchik's circumplex model.[23]

[22] The three dimensional model can be seen in the upper-left corner of image, with the two dimensional occupying the remaining space.

In the centre of the model (at the top of the cone), Plutchik represented eight primary emotions[24] as a colour wheel, with similar emotions close to each other. Opposite emotions were 180º apart from each other and were drawn with a complementary colour. Secondary emotions (the ones outside the central circle) result from a mixture of the primary and/or changes in intensity, with their colours also reflecting this process. The cone's vertical dimension (its height) represents their intensity.

For example, *grief* (blue) has *ecstasy* (yellow) as its opposing affective state, both with complementary colours. *Sadness* and *pensiveness* (also blue) are similar emotion to *grief* but with reduced intensities (also shown by the colour lightness). *Disapproval* is a secondary emotion that results from mixing *grief* and *amazement* or similar less intense emotions.

2.2.3 Lövheim's cube of emotion

Hugo Lövheim (2011) presented another tridimensional model for representing emotion. According to him, emotions are based in three monoamines: *Noradrenaline*, *Dopamine* and *Serotonin*. These are mapped to a three dimensional space, with each axis being represented by one of the monoamines. This result is a cube, aptly name *cube of emotion*, as seen on **Figure 5**.

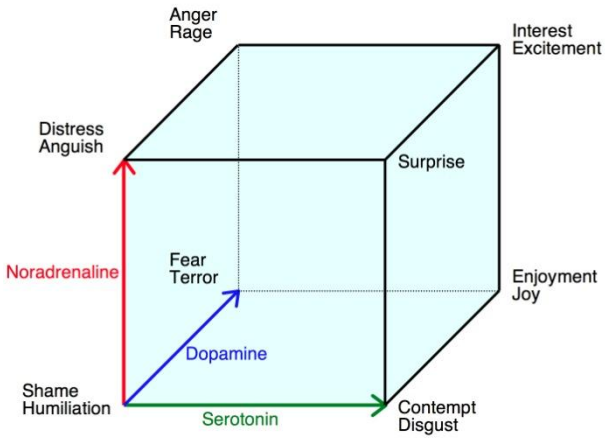


Figure 5: Lövheim's cube of emotion.[25]

According to Artur Nóbrega (2014) each vertex of the cube is mapped to a specific emotion,[26] each belonging to one of three groups: *positive emotions*,[27] *neutral emotions*[28] and *negative emotions*.[29] Each emotion is represented by two words so that the axis is able to portray

[23] Image source: www.fractal.org/Bewustzijns-Besturings-Model/Nature-of-emotions.htm (accessed February 10, 2016).
[24] These are: ecstasy, admiration, terror, amazement, grief, loathing, rage, vigilance.
[25] Image source: http://animalelectricity.blogspot.pt/2012/08/the-lovheim-cube-of-emotion.html (accessed February 10, 2016).
[26] These are *shame/humiliation*, *contempt/disgust*, *enjoyment/joy*, *fear/terror*, *distress/anguish*, *surprise*, *interest/excitement* and *anger/rage*.
[27] *Enjoyment/joy* and *interest/excitement* are included in this group.
[28] The only neutral emotion in the diagram is: *surprise*.
[29] The remaining five emotions belong to this negative emotions.

different intensities: the vertexes represents the extreme emotions, the middle of the axis a neutral value and the remaining space the different intensities between. A summary of the relation between emotions and the monoamines can be found in **Table 2**.

Table 2: Relation between emotion and neurotransmitters based on Lövheim's cube of emotion

Emotion	Serotonin	Dopamine	Noradrenaline
Anger / Rage	Low	High	High
Distress / Anguish	Low	Low	High
Contempt / Disgust	High	Low	Low
Enjoyment / Joy	High	High	Low
Fear / Terror	Low	High	Low
Interest / Excitement	High	High	High
Shame / Humiliation	Low	Low	Low
Surprise	High	Low	High

With the all the emotions models found during our research presented, it is still necessary selecting one to use throughout this work to represent affective states. In our view, Lövheim's cube of emotion is an unfeasible solution, as monoamines are found in the blood, meaning that our framework would have to have access to the players' blood to perform its functions. With two models remaining, we researched which literature had used which inside our research field. The most common has Russell's circumplex model of affect, with this one also being our selection.

2.3 Affective and biofeedback

2.3.1 Biofeedback

According to *The Association for Applied Psychophysiology and Biofeedback*, biofeedback is:

> (...) a process that enables an individual to learn how to change physiological activity for the purposes of improving health and performance. Precise instruments measure physiological activity such as brainwaves, heart function, breathing, muscle activity, and skin temperature. These instruments rapidly and accurately 'feed back' information to the

> user. The presentation of this information — often in conjunction with changes in thinking, emotions, and behavior — supports desired physiological changes.[30]

According to David Rego, biofeedback is typically used in the clinical field, as treatment for medical conditions[31] or for human performance improvement, although in the last decade it has also been applied to human-computer interaction (HCI), to improve players' experience and studying limitations in the creation of video games focused exclusively in these mechanisms(2014, 4).

Biofeedback can also be classified differently depending on *the players capacity of consciously controlling the biosignals used as input* or the *awareness of this control*.

Regarding the first, Lennart Nacke, et al. (2011) say biofeedback could be divided in *direct* or *indirect*. If the game system is using as an input biosignals that the players have conscious control of, than it is considered *direct biofeedback*. A practical example of this type of feedback is given by Vasco Torres, who suggests placing an electromyography (EMG)[32] sensor in a muscle that the player has a consciously control of, and using it as an input for the game (2013, 7). If, however, this biosignals cannot be consciously controlled by the players,[33] than it is considered *indirect biofeedback.* According to Torres, applying EMG sensors in the face muscles is an example of this type of feedback as the players is not always aware of their usage (2013, 7). By analysing which muscles are being used, it is possible to determine which facial expression are made by players,[34] thus establishing their mood.

Nacke (2011) warns us that the distinction between both is unclear as an indirect input could suddenly become a direct input. For example, breathing is a natural process that humans do not need to control consciously and happens unconsciously . However, at any moment, we can take conscious control of it, manipulating our breathing as we desire.

Regarding *awareness of control*, Kai Kuikkaniemi et al. (2010) state the *biofeedback* could be either *explicit*, where players are aware of their ability to directly control the game through their biological signs, or *implicit*, where players' biosignals are still being used by the game although they are not aware of this fact. Kuikkaniemi also warn us that the distinction between both is somewhat unclear, as at any moment, the players may suddenly become aware of the inner workings of the system and be able to control it, making implicit turn explicit biofeedback.

[30] Definition by The Association for Applied Psychophysiology and Biofeedback, Inc.. Source: http://www.aapb.org/i4a/pages/index.cfm?pageid=3462 (accessed June 30, 2016).
[31] Such as migraines and incontinence (Gilleade, Dix and Allanson 2005, 2).
[32] For more information on the type of available sensors, the reader is directed to section **2.3.4**.
[33] The players' heartbeat would be one example.
[34] Frowning, smiling and surprised are examples of facial expressions that can be captured using this process.

2.3.2 Affective computing

Affective computing is a term coined Rosalind W. Picard, in 1991, to refer to "computing that relates to, arises from, or deliberately influences emotion", and is an area of human-computer interaction, focusing on emotional communication and "the appropriate means of handling affective information." (1)

2.3.3 Affective feedback

According to Daniel Bersak et al. (2001),[35] *affective feedback* is a junction of *biofeedback* and *Affective Computing*. The authors present an example of this type of feedback where they developed a two player racing video game, called "Relax-to-Win," where players compete against each other by relaxing. The systems read the players stress level, through their galvanic skin response,[36] and made their in-game dragon avatar move faster or slower depending on this value. The winner is the player who managed to stay relaxed the most time during the game.

Kiel Gilleade, Alan Dix and Jen Allanson present their definition of *affective feedback* at the same time they distinguish it from *biofeedback*:

> What is distinctive about the notion of affective feedback in comparison to biofeedback is that the physiological changes in the loop are uncontrolled. In biofeedback games the player explicitly participates in controlling their physiological responses in order to control the game world. In contrast the player may not even be aware that their physiological state is being sensed during play of an affective videogame as the intention is to capture their normal affective reactions. (2005, 3)

> This form of gameplay is commonly referred to as affective gaming; where the player's current emotional state is used to manipulate gameplay. (2005, 2)

Based on this, *affective feedback* can be seen as implicit biofeedback that uses the captured information to determine the players affective state and, from it, shape the game. Also, Ruben Aguiar (2014) refers to affective feedback as *affective gaming* and Gonçalo Silva (2014) refers to it as an area of Affective Computing.

Based on the terms presented, our work will position itself in all the areas described before because: 1) The framework we developed allows for the receiving of several biosignals[37] each can be used as an input for controlling the game (thus *biofeedback*); 2) the information received will also be used to extract the players' affective state, which in turn will be used to affect the

[35] Who is also the coiner of this term (Gilleade, Dix and Allanson 2005, 3).
[36] More details on this type of biosignal, and to measure it, will be explained in section **2.3.4**.
[37] With their number and type defined by the game developer and designer.

game (thus *affective feedback* or *affective gaming*). This work is also inserted in the area of *affective computing*, seeing as *affective feedback* is component of the said.

2.3.4 Types of biometric data

There are several types of biosignals that can be read from the player. Based on Ambinder (2011), Rego (2014) and Torres (2013), the following physiological data can be collected for evaluation.

2.3.4.1 Brain activity

Brain activity can be seen as the attention level (Rego, 9). According to Rego, the frequency of the brain can generally be associated to a state of attention, as seen in **Table 3**.

Table 3: Frequency of the electrical activity of the brain and its respective interpretation. (Rego, 9)

Frequency	14-30 Hz (beta state)	8-12 Hz (alpha state)	0.5-3.5 Hz (delta state)
State	High level of attention	Relaxed or moderate attention	Drowsiness

Brain activity is collected through sensors capable of performing an electroencephalography (EEG).[38] It is tied with indirect feedback since this biosignal is not something that the player can control willingly (Rego, 9).

The advantages are that it gives a good measure[39] of the player's arousal and emotional valence (Rego 2014, 9). However, these sensors tend to be expensive, intrusive, easily influenced by external noise and the data provided is difficult to be endorsed (Ambinder, 26).

2.3.4.2 Breathing

Breathing can be used to measure the number of inspirations and expirations the player makes and how deep they are. Torres explains that this type of sensor is commonly a stretchable belt, tightened around the thoracic or abdominal area (2013, 10).

Since breathing is an unconscious action, it can be tied to indirect feedback. However, it can also be easily controlled, placing this biosignal in the direct feedback simultaneously (Torres, 10).

[38] An analysis that measures the electrical activity of the brain.
[39] So long as there is no noise (interference) affecting the collection of data.

2.3.4.3 Eye movements and pupil

The eye is also a source of biometric data. By analyzing its movements, dilation and fixations, one can extract information such as the player's measure of attention and arousal (Ambinder, 23).

We assume that eye movements is a biosignal belonging to both categories (direct and indirect) of feedback as it is consciously controllable although, in certain scenarios,[40] this control might escapes us. The eye pupil, however, is indirect, as its actions are controlled by autonomic nervous system.

This data is normally captured trough the use of camera (which might attract the attention of the players, biasing their eye movement) and an eye tracking software. One negative point is that this software is expensive and requires extensive analysis (Ambinder, 23).

2.3.4.4 Muscle tension

Muscle tension expresses the state of a body muscle. An electromyography (EMG) is a process which reads the electrical activity of the muscle or muscles under analysis and determines if they are being contracted or relaxed (Rego, 10). This biosignal is more efficiently used in direct feedback, as the players are able to consciously control many of their muscles, although it can also be applied to indirect feedback, as some muscles escape this control under certain circumstances[41] (Torres, 7).

One of the advantages is that it can be used in several body parts and the measurement is easily performed. On the other hand, these type of sensors tend to be expensive, although cheaper alternatives are beginning to surface, and their placement might be intrusive to the player(Rego, 10).

2.3.4.5 Facial expressions

The players's *facial expressions* can also be used as biofeedback input since they often portray their current disposition. This biosignal can give us the level of the valence, arousal and "measure instantaneous responses"(Ambinder 2011, 20).

There are two possible ways to acquire it: through muscle tension or through the use of a computer vision algorithm. Regarding the first option, it is possible to perform an electromyography on the players's facial muscles, thus ascertaining which ones are being used and which are not. From there, it is possible to determine which facial expression the players are displaying and , from it, their current mood (Torres, 7). Once again, the advantages and disadvantages of this method are the same as the ones described in the previous section (section **2.3.4.4**).

[40] Humans unconsciously close their eyes when an obstacle draws near due to the reflex arc, a mechanism that controls our reflex actions.

[41] When frightened, the muscles in the human face automatically contract and relax in a certain pattern, forming scared facial expression, even if only for a brief moment.

The alternative way is to use a computer vision software to perform this analysis. Typically, the process is similar to the one used in the capturing of the eye movements and pupil biosignals: a camera captures the player face and a software performs the necessary operations to detect the players's face and determine their facial expression. However, this is often expensive and some algorithms require training in order to work properly (Ambinder, 20).

In regards to biofeedback, facial expressions are similar to muscle tension, belonging to both direct and indirect feedback.

2.3.4.6 Heart rate

Heart rate represents the frequency at which the heart is beating. There are two possible ways to determine this value: through an electrocardiography (ECG) or through the player's blood pressure, both performed by sensors. This biosignal is used in indirect biofeedback as the heartbeat is a phenomenon that the player is typically unaware.

An ECG is a process that reads the electrical activity of the heart through the use of electrodes placed on the player's skin across the thorax or chest. These type of sensors are able to determine the players' arousal index, are not expensive, measures are easily performed (Rego, 11). However, as the heart rate takes some time to reflect the stimuli[42] and so does the data captured (Ambinder, 15).

Blood pressure is the strength with which the blood pressures the inside of the blood vessels[43] as it is pumped by the heart, and can be used to determine the heart rate. These sensors are also able to detect the players' arousal level, have accessible prices but their values suffer a lot of variations (Rego, 15).

2.3.4.7 Skin conductance level

Skin conductance level, also known as *galvanic skin response* (GSR) or *electrodermal activity*, represents the electrical conductance (or resistance) of the skin and is directly related to the production of sweat. According to Pejman Mirza-babaei et al. the skin acts as a variable resistor (2011, 4), hence the more nervous the player becomes, the more sweat is produced, the better electrical conductance becomes.

These sensors are typically placed on the tip of the fingers, although they can be placed in other body parts, are not expensive and are able to determine the player's arousal level (relaxation level). Regrettably, it's difficult to determine the valence index as is to associate the values with the events that motivated them (Rego, 13).

In regards to biofeedback, this biosignal is considered indirect feedback has the player as no control of this biological behaviour: the sympathetic nervous system is the one responsible for controlling it.

[42] If players are scared their heart rate will only rise after the stimuli is perceived and not immediately or before it happens.
[43] Source: http://www.bloodpressureuk.org/BloodPressureandyou/Thebasics/Bloodpressure (accessed June 3, 2016).

2.3.4.8 Temperature

Temperature can be defined as "a measure of the average kinetic energy of the particles in a sample of matter, expressed in terms of units or degrees designated on a standard scale."[44] However, more commonly, it is used to refer to "the degree of hotness or coldness of a body or environment,"[44] which is the definition we will be using.

In regards to humans, this biosignal is the temperature of the human body. Depending on the body part this type of sensor is applied to, values may change. Temperature is another possible alternative for establishing the arousal level of the player(Rego, 14).

One of its advantages is the affordable price of these sensors, which are capable of ascertain a good arousal level although read values suffer a lot of variations(Rego, 14).

2.3.4.9 Other biosignals

Other biosignals that could be used as biofeedback input are: *blood analysis*, *oxygen*, *posture* and *gestures*.[45]

Blood analysis consists on evaluating the components of the players' blood while they are playing. By knowing which elements are present in the blood and their quantity it would be possible to determine the players' affective state.[46] In terms of biofeedback, we believe that this biosignal could be used in implicit biofeedback as the players are unable to change their blood constitution whenever they desire.[47]

Oxygen saturation in the players' blood[48] is also a biometric value. Thankfully, a different measuring process where direct access to the player's blood is not required exists, called *oximetry*.

Posture is the position of the player's body or body parts. According to Ambinder, this biosignal could be used to determine the players' valence index and, if combined with pupil dilation, could also be used to determine their frustration (2011, 27). We assume that posture can be used in direct feedback as the players can easily manipulate their posture and body parts. Similarly, gestures are movements performed by body parts and can also be used as biometric input.

2.3.5 Ways of collecting biometric data

Biometric data is exclusively collected through sensors that vary by type of biosignal captured (some may capture more than one type), quality, technology used, among other factors.

[44] Source: http://www.thefreedictionary.com/temperature (accessed June 3, 2016).
[45] Rego refers to voice recognition as biosignal but does not expand in what are its advantages or disadvantages. In lack of this information, we decided not to include it (2014, 8).
[46] Lövheim's cube of emotion uses the quantity of dopamine, serotonin and noradrenaline present in the blood to determine the subject's current mood.
[47] This, however, could be achieved through feeding, drugs and others techniques. Nevertheless, we will assume that this would be impossible while the player is focused on the game.
[48] Determined through blood analysis.

Some sensors might require a decoder[49] since they are unable to convert the captured signal into information intelligible to their receivers.

As one of the objectives of this work was developing a framework able to capture the information sent by different sensors, at some point in time we would require one or more sensors to test this functionality. To discover which kind of sensor were better suited to our work, we performed a research and compiled our finds in a list, which the reader can find in **Collection of biometric sensors**. We ended up choosing Plux BITalino, as it offered a set of different sensors in one package.[50]

2.4 Other frameworks for biofeedback.

While searching for solutions that could be used to develop our framework, Pedro Nogueira et al. (2013)'s framework proved to be interesting asset to us. This framework is named Emotion Engine (commonly referred by its acronym: E^2) with its structure present in **Figure 6**. The system is divided in six components: *Physiologically-Inductive Emotion Recognition Sub-system* (PIERS), *Affective Reaction Extraction and Extension Sub-system* (ARE^{2}S), *Affective Reaction Compendium* (ARC), the *Closed-Loop Emotional Adjustment and Regulation Sub-system* (CLEARS), *Game Layer alteration Daemon Operating Script* (GLaDOS) and *Implicit Mechanism Pool* (IMP).

[49] Decoder, in human terms, is something akin to a translator. What it does is receive a raw signal, processes it and transform it to a format (typically a number) understandable by its receiver.
[50] More information on this sensor can also be found at **Collection of biometric sensors**.

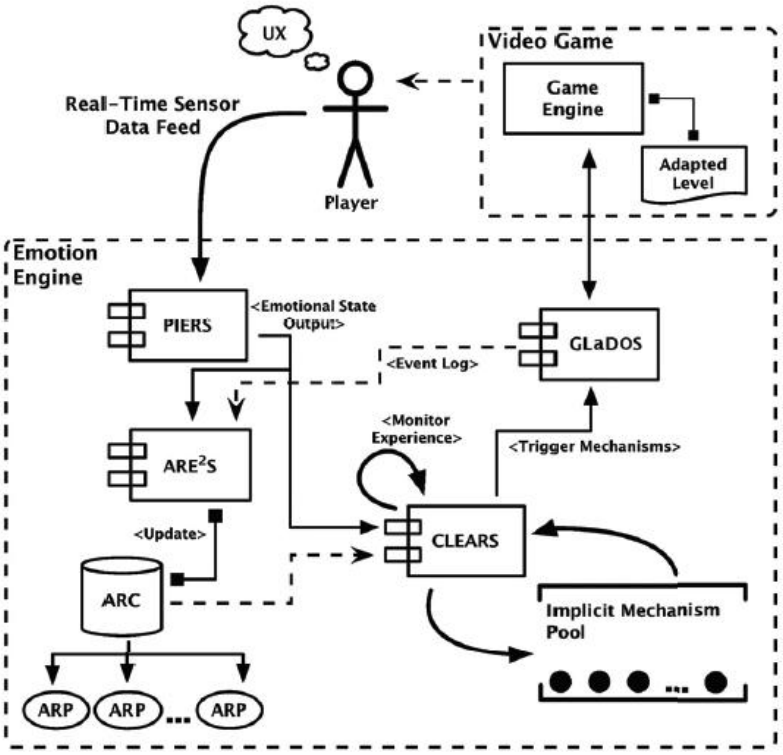


Figure 6: Nogueira et al. (2013) Emotion Engine structure.

PIERS is responsible for determining the user's current emotional state, classifying it using Russell's arousal and valence space. The system is able to transform the sensors[51] data into the aforementioned values by filtering them through a two-layered classification process. This process is summarized in **Figure 7**. Nogueira also informs that this process has achieve an accuracy rating of 85% for arousal and 78% for valence.

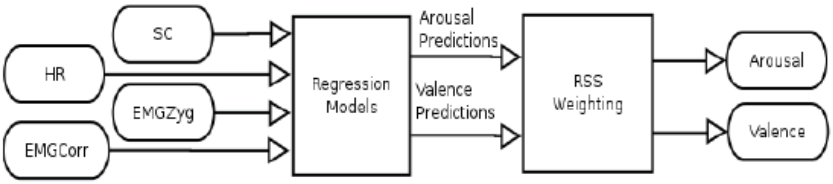


Figure 7: PIERS inner-workings. Source: (Nogueira, et al. 2013)

ARE²S is responsible for establishing a connection between the game events and the emotional state of the player. To perform this task, it receives a log from GLaDOS and the affective state from PIERS. The results are stored in ARC, which is a database containing the all the *affective reactions profiles*[52] (ARP) of each player.

[51] The sensors used measure the skin conductance level (SC), heart rate (HR) and the muscle tension (EMG) of the Zygomaticus Major (cheek) and Corrugator Supercilii (brow) (Torres 2013) .
[52] How the player reacted to a game event.

CLEARS is responsible for monitoring the player's emotional state and trigger certain events, which are then sent to GLaDOS. These events have the purpose of bringing the emotional state of the player to the intended state defined by the game developer or designer and are stored in IMP.

GLaDOS receives the events from CLEARS and communicates with the game in order to affect it (perform the actions stated in the events). It is also responsible for receiving the game events and work in tandem with ARE^{2}S.

Of the modules described, PIERS would prove to be a powerful asset if implemented in our work.

2.5 Profiling

With the biofeedback input available in a video game, it would be interesting to profile the players' affective behaviour during the game and use them to adapt their playing experience. This way, it would be possible to determine which kind of affective states are evoked by certain in-game events and categorize the player depending on their affective reactions..

Profiling is a term used in a variety of scientific fields, such as psychology, user experience, game design, among many others. Framing it specifically within the field of game design, Pedro Cardoso presents a definition for the term:

> Profiling essentially aims at the discovery of patterns in data in order to identify or represent something or someone, be them individuals or groups. While profiling, the system analyses a player's behaviour and interprets the emerging patterns in order to establish a course of action. Profiling is not just about activating or collecting particular objects, or accomplishing quests. It features a much deeper and complex design. It is about what collecting that object or accomplishing that quest means. It is about understanding what it means to undertake tasks, about understanding how the player plays the game, how she accomplishes a certain goal or how she acts in a given situation, throughout a specific section or even for the duration of the entire game. It is about interpreting behaviour, analysing sets of actions and understanding even the most subliminal behavioural patterns, and acting based on that. (2015, 277-278)

According to Cardoso, one needs to be aware of three things in a profile system:

1. The player's awareness that he/she is being profiled;
2. How is the profiling being used to alter player experience: to balance or unbalance the game;
3. The *player's play history*.

In terms of player awareness, profiling can be divided in two types: *explicit* and *implicit*. *Explicit* indicates that the players are indeed aware that the system is using their actions and behaviours to perform the profile, and they also know how to manipulate this information in order to direct the game where they intend; while *implicit* is the opposite: the players do not know they are being profiled or, if they do, they do not know how to influence the game because they lack an understanding of the processes behind it.

Cardoso presents the game *Silent Hill 2* (2001) as an example where profiling is used.[53] The game possesses three possible endings, each directly related to one of the three categories of player profiles (one profile per ending) that the system recognizes. By analyzing the player's actions, behaviour and decisions taken while playing, the system decides which category the player belongs to, unlocking the respective ending.

To achieve the first ending[54] the players must, while playing, manifest a behaviour that shows a sense of self-esteem or self-preservation and avoid interacting with a certain game character that represents a possible new lover for the players' avatar (2015, 279-280). On the other hand, to attain the second ending the players need to contradict some of the actions of the first, namely start cultivating their relationship with that new character. The necessary behaviour to achieve the third ending also contradicts the previous two but on different conditions. This time the players must stop showing the sense of self-esteem or self-preservation: as if they were depressed and close to committing suicide (2015, 280).

Based in these rules, the game system is able to evaluate how the players behave and act, categorizing them into one of three possible profiles, and presenting them the end that matches their profile. There is, however, an important point to note in this process: first time players will probably be unaware of the profiling done by the game. This makes *Silent Hill 2* 's profiling *implicit*. Now, if the reader, knowing the existence of profiling and the set of rules operating it, decided to play this game, the reader would know exactly which behaviour to emulate to reach a specific ending thus making the game system's profiling become *explicit*.

Implicit becomes explicit when the players are informed of the profiling inner workings or when they gain the understanding of the system by themselves and are able to influence it.

Profiling can also be used with two purposes: *to balance* or *to unbalance* the game. While playing, players' may get frustrated or bored, as these affective states arise whenever the game's difficulty does not match the their capacities. If they are unable to clear the game's challenges with their current skills, frustration is created (the game is too hard); if the game's challenges are

[53] This example as been adapted to also demonstrate the *implicit* and *explicit* profiling, although in the original work it was used as a general example of profiling.
[54] We will only present a summary of these conditions.

easily completed and the players do not feel challenged, then boredom may be created (the game is too easy).

One important point to note is that *balancing* and *unbalancing* are not synonyms for *making the game easy* and *making the game hard*, respectively. They depend on the opposing relationship between the player, who is trying to clear the game, and the game system, who is trying to challenge the player.

Imagine the following example: a player is currently trying to overcome a challenge provided by a game but without much success. In order to avoid frustration, some elements of the challenge would have to be altered to give the player a better chance of conquering it, thus balancing the game. However, if the player intention was to face the biggest challenge possible then, in order to balance the game, the elements of the challenge would have to be changed in order to make it even more difficult to clear. With this type of player, if the challenged was made easier to complete then the game would become unbalanced. This is why this type of profiling is often called *dynamic game difficulty balance*.

Cardoso presents *Left 4 Dead* (2008) as an example where this category of profiling is used. The game possesses an artificial intelligent agent,[55] aptly called 'Director',[56] which is responsible for procedurally altering certain game elements such as placement of enemies, weapons and items.

> *Left 4 Dead* (2008) is a game that cannot be overcome by memorizing the locations of the enemies in an attempt to anticipate their moves, because the game system dynamically adjusts their presence according to the performance of the player. (...) This is an element of surprise that creates a certain unease and novelty upon each play-through, thus unbalancing the game. (2015, 284)

Ambinder also presents one example where this category of profiling is partnered with biofeedback. In this experiment, an improved version of the A.I. agent previously mentioned[57] would now consider the players' actual arousal level when making decisions. Previously to the addition of the biofeedback, the agent estimated the players arousal based on the game events triggered by the player and assumed that this value would decreased as time went by, unless a

[55] Artificial Intelligence (A.I.), according to Techopedia, is "an area of computer science that emphasizes the creation of intelligent machines that work and react like humans." Source: https://www.techopedia.com/definition/190/artificial-intelligence-ai (accessed June 5, 2016).
Source: https://www.techopedia.com/definition/190/artificial-intelligence-ai (accessed June 5, 2016).
An artificial intelligent agent is an autonomous piece of software that is able to perceive the environment it is in, through sensors, and act upon it, through effectors (also known as actuators), in order to achieve an objective or fulfil a task.
For a more detailed explanation on the concept, the reader is recommend the following link : http://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_agents_and_environments.htm (accessed June 5, 2016).

[56] The 'Director' is an A.I. responsible for overseeing the players' actions and taking the necessary steps (place enemies, alter the enemy performance, among others) to assure the best player experience.

[57] *Left 4 Dead 2* (2009), a direct sequel to *Left 4 Dead*, had several improvements implemented, one of which was the changes done to its A.I. agent (renamed 'Director 2.0') which is now able to dynamically change small elements of the map layout.

new event was activated. By using the skin conductance, the agent can extrapolate the correct arousal and balance or unbalance the game (2011, 35-40).

The experiment's results suggested that the measured arousal (value based on skin conductance) produced a greater enjoyment than the estimated arousal (estimated value based on events triggered) and it was also possible to determine, although rudimentarily, which events elicit enjoyment (2011, 49-50).

The last category of profiling is dependent on the player's play history, or more precisely, *on how complex the profiling mechanism is*. This complexity could come from how the profiling operates or from how dense the collected data is. There are two possible categories: *shallow* (short play history) and *deep* (long play history).

For *shallow* profile, Cardoso presents *Super Mario 3D World* (2013) as an example. In it, the number of times the player failed to complete a level is considered an element of the play history. Once this number is higher than five[58] the game awards the player a *power-up*[59] to ease the difficulty of the level. There is no need to collect any more data: "the actions are simple and the consequences straightforwardly implemented" (2015, 285).

For *deep* profile, Cardoso presents a demo of the game *Metal Gear Solid V: The Phantom Pain* (2015) as an example, since the game was unavailable (still in production) at the time. However, at the time of writing of our work, the game is already available in the market, so the profiling mechanism can now be validated.

One interesting point highlighted by Cardoso is that the system is able to track the player's play style and strategies, trying to counter them. Daniel Hindes confirms this fact in his review article of the game: "In an attempt to prevent you from employing the same tactics in every mission, enemies will equip themselves with gear to counter your play style. Land enough headshots, and they'll wear bullet-proof helmets. Repeatedly call in air support, and they'll start wielding missile launchers."[60]

As seen, the game profile system collects a wide range of information (areas visited, weapons used, tactics performed) from the player's actions throughout the whole level and, if a common pattern in the player's actions is found, it applies the necessary changes to void these strategies. The data collected from the player and measures implemented by the game system are also more complex than those in the *Super Mario 3D World* example: "the outcomes of *deep profiling* stem from more algorithmic complexity, interrelating the obtained data in order to

[58] When the player had to retry the same level for the sixth time.
[59] *Power-up* is a lingo used to refer to objects that, when collected, give a set of benefits, bonuses or other special powers to the avatar, *powering* him/her *up*.
[60] The review article can be found at: https://www.rockpapershotgun.com/2015/09/14/six-things-mgsv-the-phantom-pain-could-do-better/ (accessed April 7, 2016).

analyse the behaviour of the player and to identify patterns, which is something that may require a longer play history." (2015, 287).

Before concluding, however, it's important to mention that a profiling technique can possess diverse traits. The profiling done in the game *Super Mario 3D World* is a perfect example of such. As shown before, based in the play history, it is clearly *shallow profiling*. However, the solution given to the problem (create a *power-up* to help the player clear the level) shows that this type of profile has the effect of *balancing* the game, when looking at it from the balance/unbalance perspective. Also, as this is a mechanism whose rules are not difficult to understand and can be easily detected, especially for players who replayed a level several times, this makes this type of profile *explicit*, in terms of player's awareness. **Table 4** presents a summary of the profiling traits.

Table 4: Profiling traits

Profiling categories		
Player's awareness	Explicit	Implicit
Effect in the game difficulty	Balance	Unbalance
Player's play history	Shallow	Deep

Finally, it's also important to mention the reason why *profiling* was included in the design of our framework was because we believed this tool could be an powerful asset to game developers and designers, as it would allow them to study and analyse their players' affective state during gameplay; or be implemented as a game mechanic, similar to the examples presented, to improve the players' experience while playing.

2.6 Dramatic tension in games

Dramatic tension is one important aspect of game development and design. So important that several mechanics have been created and used in order to control it with the purpose of assuring that the players are engaged and challenged (and, as result, having fun).

This section draws heavily from *Tools for Creating Dramatic Game Dynamics* (LeBlanc 2005), where the author presents his definition of drama in games, details the components that are necessary for it to occur and several techniques to make it happen. Many of these techniques

were used as a foundation for some of the proposed game mechanics in this work although with the necessary changes so that the biofeedback could infused into them.[61]

For Marc LeBlanc dramatic tension is "our level of *emotional* investment in the story's conflict: the sense of concern, apprehension, and urgency with which we await the story's outcome" (2005, 443), often being the primary motivation for playing (2005, 439). Drama originates from *conflict* which takes the shape of a *contests* in games.[62] These are challenges that test the players' wits and/or physical prowess, sometimes faced alone[63] or between multiple players[64] (2005, 444). Unfortunately, drama cannot be created or controlled directly in games: the game designer "can only create the circumstances from which the drama will *emerge*" (2005, 440).

For dramatic tension to exist two elements must be present: *uncertainty* and *inevitability*. Individually, they are unable to generate dramatic tension; both must be present for that to happen.

Uncertainty is the players' lack of knowledge on the outcome of the challenge they are facing. It is not possible, at the beginning, to know if the player is going to win or lose; it is not possible to know which player is going to be the victor or the defeated. In other words, the outcome is *uncertain*. For example, in a game where the score of both players is very close or tied, it is difficult to determine who is going to win, thus uncertainty is created.

To create this element three techniques can be used: *force*, *illusion* or *a mixture of both*. *Force*, as the name may suggest, manipulates the state of the game or its variables, limiting the advantage one player has over the other, making the game more fierce.[65] *Illusion* manipulates the players' perspectives in order to make the contest look more closely though than it really is.[66]

One mechanism for creating uncertainty, using the *force* technique, is *feedback systems*. Feedback systems are composed by:

1. a *game state*, which contains the "complete status of the game at the particular moment" (2005, 447). For example, in Checkers,[67] LeBlanc claims that the game state would

[61] These game mechanics will be presented and described in section **3.1.8 Manipulate the dramatic tension in a game**.
[62] This shape changes depending on the medium (books, videogames, game boards, etc).
[63] A single player game.
[64] A multiplayer game.
[65] A quick example would be a racing game. *Force* would increase the losing player's car speed, making it possible to catch up with the winning player vehicle.
[66] The turbo mechanic is one example of illusion in racing games. When the losing player uses the turbo to overtake the winning player, depleting it completely, this player now becomes the first. However, this is not more than an misinterpretation as the other player still has his/her turbo making it possible to return the game to its previous state if he/she so wishes.
[67] *Checkers*, also called *Draughts*, is board game played between two player, sharing some similarities with *Chess*. Each player possess twelve checkers (round circles), with each set of twelve possessing the same color (normally,

contain information on the checkers' position on the board, as well as the next player to move. This information could be expanded to also contain which checkers are crowned, for example;

2. the *scoring function*, which is "a rule of the game that gives us a numerical measurement of who is winning and by how much" (2005, 447) and works differently from game to game. In a racing game the score is based on the distance between each car[68] while in a *platform game*[69] the score might be given by the number of enemies destroyed; One other important property is that "A good scoring function produces a large number the greater the winning player's lead, and produces the number zero when the game is tied" (2005, 447).
3. the *game mechanical bias*, which is "a rule of the game that gives one of the contestants an advantage over the other" (2005, 448), also varying from game to game. As an example, in racing games, this could happen by letting a player start the race a number of seconds before the other; in fighting[70] games, assigning less health to one of the players would constitute a bias;
4. the *controller*, which is "a rule of the game that chooses which player receives the game mechanical bias" (2005, 447).

The feedback system could be subdivided in two types: the *negative feedback system* and the *positive feedback system*. In the former, the game mechanical bias is applied to the losing player giving him/her an advantage,[71] *forcing* players into a tie, thus creating uncertainty. Leblanc presents the example of a racing game where the bias is an increase in the maximum speed of the car and the scoring function the distance between the two cars. When the distance between the two exceeds a certain value (when the system considers that the vehicles are far away enough), the bias starts working, allowing the loser to catch up with the winner.

Positive feedback systems do exactly the opposite: they apply the bias to the player who already had the advantage,[72] making him/her even more likely to win. This, however, reduces uncertainty, which would make these systems a inadequate method for the task at hand. To this point LeBlanc argues that:

---

black or white) so to able to distinguish which checker belongs to whom, placed in a board of 8 x 8 squares. The objective of each player is to destroy (remove from the board) the adversaries checkers while keeping his/hers alive (avoid having the checkers remove from the board).
A quick overview of the game, rules and other technical aspects can be found at: https://www.youtube.com/watch?v=SuQY1_fCVsA (accessed March 30, 2016).

[68] This value is sometimes presented as meters/yards, other times as seconds.

[69] *Super Mario Bros.* (1985) is one example presented as a platform game. An excerpt of the game being played can be found at: https://www.youtube.com/watch?v=ia8bhFoqkVE (accessed March 30, 2016).

[70] *Street Fighter II* (1991) is presented as an example for a fighting game. An excerpt of the game being played can be found at: https://www.youtube.com/watch?v=6OlenbCC4WI (accessed March 30, 2016).

[71] This advantage makes the scoring function tend to zero.

[72] In other words, the player who was already in first place receives a boost.

> Toward the end of a game, positive feedback systems are sometimes useful for dispelling uncertainty, bringing about the climax, and creating a sense of finality and closure. Sometimes the negative feedback systems in the game can cause the game to stagnate. Positive feedback provides a mechanism for breaking the equilibrium and moving the game forward. (2005, 449)

As seen, these systems reduce uncertainty because they bring closure and rectify stagnant situations. As such, they should be carefully used in order to avoid breaking the dramatic tension before the right moment. However, uncertainty, and consequently dramatic tension, cannot go on forever because they risk killing the players' enjoyment, whence the importance of positive feedback systems.

The mechanism that resorts to *illusion* techniques for creating uncertainty are: *Escalation*, *Hidden Energy*, *Fog of War*, *Decelerator* and *Cashing out*. However, we will only focus on the third.[73]

Fog of War is a military concept, coined by the military analyst Carl von Cluasewitz, and refers to "the uncertainty in situational awareness experienced by participants in military operations. The term seeks to capture the uncertainty regarding one's own capability, adversary capability, and adversary intent during an engagement, operation, or campaign."[74] This concept is commonly used in games, especially in strategy games. According to Leblanc, *Fog of War* "simulates limitations of game characters' abilities to perceive and monitor the world around them. The 'fog' covers all parts of the map that the players' units cannot see" (2005, 451).

As an example, we present the game *Sid Meier's Civilization III* (2001), with the *Fog of War* mechanic visible in **Figure 8**. In it, it is possible to distinguish three areas: region 1, a space where the landscape and game units are observable by the players without obstructions; region 2, similar to region 1 but slightly darker; and region 3, completely pitch black. Through the use of the Fog of War mechanic, the process of collecting tactical information from the environment is hindered. In region 2 the players are unable to perceive the presence of enemy units in the area and in region 3 it is even impossible to observe the landscape.[75] This information could prove to be useful when making decisions. By limiting the number of information available to the player uncertainty is then created.

[73] The decision to focus only on the Fog of War mechanics is made based on its contribution to the game mechanics present in section **3.1 Uses of biofeedback in video games.** In no way is intended to debase the remaining as they also contain interesting mechanics for manipulating the dramatic tension.
[74] Definition taken from The Free Dictionary, by Farlex: http://encyclopedia.thefreedictionary.com/Fog+of+war (accessed April 3, 2016).
[75] As such, information about the resources presented in the area are also hidden.

Figure 8: The effect of the game mechanic 'Fog of War'.[76]

Inevitability is the knowledge that the contest has an end and is surely moving towards it. The conclusion must be *inevitable*, otherwise the outcome of the conflict will seem distant, giving players little incentive to invest their emotions in the contest (2005, 445) as there is no sense of urgency, resulting on the dramatic tension being dispelled (453).

Inevitability is created by using the concept of a 'ticking clock': any mechanic that is able to give a "measurement of their [player] progress through the game, as well as a sense of how far away the end might be" (2005, 453). The most basic example is an in-game clock informing the players how long they have to complete the current challenge. This mechanic is often present in many games of the *Super Mario Bros.* franchise, where a timer is featured in the upper right corner of the screen (**Figure 9**). If the players are unable to complete the level before this counter reaches zero, they fail the level and have to try again.

Figure 9: The "ticking clock" mechanic (upper right corner). Screenshot from *New Super Mario Bros. U* (2012).[77]

[76] Screen capture of the game *Sid Meier's Civilization III* (2001). Source: http://www.gameology.org/screenshots/civ_iii_fog_of_war (accessed April 3, 2016).

LeBlanc presents several other examples where the game mechanic is not a clock/timer, yet having the same effect: "the increasingly crowded game board", "the waning deck sizes", "the decreasing health bars", "the depleting gold supplies",[78] the gradual collection of clues in games such as *Clue*,[79] the number of remaining meters/yards until the end of a race, among other mechanics (2005, 447).

It is also important to note that this mechanism should also have two more characteristics: being *irreversible* and *perceivable*. The former means that changes performed by the mechanism cannot be undone[80] otherwise the players might feel that the end of the game is avoidable, and thus inevitability is lost. One interesting example where this inevitability is dispelled is found in *Super Mario Bros. U*: the ticking clock on the upper screen cannot be rewound however, in some levels, certain items exist that permit time to be recovered, whence dispelling inevitability.[81] The latter means that the player must recognize the 'ticking clock' as one and understand how it operates; a secret or a far too complex the mechanism will fail to fulfil its role.

Finally, it is important to mention that, over the course of the game, the intensity of uncertainty and inevitability is expected to differ: *inevitability* should increase as the game progresses (the ending is coming, becoming more apparent as the game advances) and *uncertainty* should be high in the beginning and start to disappear as the climax is approaching. The climax happens "at the moment of realization: the moment when the outcome of the contest is known, and the uncertainty has been dispelled" (2005, 445).

2.7 Summary and conclusions

In this chapter we found a definition for the word *emotion*, a core concept in our work, having choose Droog's definition of *affective state*, as it encompassed the different terms that typically (although incorrectly) are used as emotions synonyms. We also detail three models used to represent affective states, from which we choose Russell's circumplex model of affect as it was the most commonly used, in the read literature.

---

[77] Screenshot source: http://venturebeat.com/2016/03/10/all-17-super-mario-bros-games-ranked/ (accessed April 3, 2016).
[78] Or similar resources.
[79] *Clue*, also known as *Cluedo*, is a game where the players must discover who, how and where a murder was committed. The culprit, murder weapon and place are randomly selected. For a quick overview of the game, the reader is recommended to watch the following explanation of the game: https://www.youtube.com/watch?v=EPjkXLksyhA (accessed April 3, 2016).
[80] This means that time should not be resettable, card decks should not be refilled, health should not be recoverable, gold supplies should not regenerate, among others.
[81] It is also possible to look at this items as an example of the *illusion* mechanic, namely the *hidden energy* mechanic. While they look like an extension of time to the player, from the designers point-of-view this time was already taken into account when developing the level, creating the illusion that the player is getting extra time when, in fact, this time has just hidden from then.

Regarding feedback, we determined the distinction of affective computing, affective gaming, biofeedback, some concepts important to our work but which we could differentiate at first. We also researched how the different types of feedback are applied and discovered that they resort to different types of sensors, each able to capture one or more types of biosignals. We then proceeded to investigate which sensors were available in the market, choosing to use Plux BITalino in our work.

Our research also lead us to search for solutions or frameworks with a similar intent, so that we could use or adapt them to our work. We detailed Nogueira's as it offered one component that could be helpful for us when developing our work.

We conclude this chapter by focusing on profiling, a tool which we believe will bring several boons to game developers and designers when joint with biofeedback, and dramatic tension in games, from which we extracted some ideas for the game mechanics presented in the coming chapter.

Chapter 3

Conceiving a framework for the manipulation of video game elements using the player's biometric data

In this chapter we will focus on the description and conceptualization of a framework able to support video games where developers and designers wish to use the player's biometric data. This chapter will be divided in two sections, in which the first contains a compilation of several game mechanics, found in concrete examples. We also propose the alteration of certain common game mechanics so that they allow biofeedback, when previously they did not.

In the second half, we will then describe the concept of the framework: a system able to capture the players' biofeedback and use it to influence the video game employing it. This framework is composed of four elements: 1) the component responsible for capturing and processing the biosignals; 2) a rules system that will receive them and translate them into actions or instructions; 3) a game which will receive these and change its state to match the players' affective state; and 4) the player.

3.1 Uses of biofeedback in video games.

From the research performed, it was determined that biofeedback could be used to influence diverse aspects of a game, being possible to categorize these in nine different groups depending on its effects. These are:

1. Influence or change the game aesthetics;

2. Dynamically adapt the players' actions and performance;
3. Complement the game's controller as a form of input;
4. Control the player avatar's behaviour, precision and performance;
5. Control the elements that act in opposition to the player;
6. Control the game's difficulty;
7. Change or adapt the game's narrative;
8. Manipulate the dramatic tension of the game;
9. Generate the game level layout.

The following subsections will address in more detail each group while presenting some examples.

3.1.1 Influence or change the game aesthetics

An example can be found in the video game *Nevermind*[82] (2015), game that according to its creators is the first commercial game to use biofeedback.[83]

In it, the players' stress level is used to influence certain elements of the scenery in which they are currently located and to add visual effects their vision[84] as can be found in **Figure 10**. From left to right, the players' stress level is gradually increasing (with left example having the lowest level) leading to the appearance of a visual interference, something akin to mist, white in colour, reducing the players' visibility. On the rightmost example, players are in such way stressed that their visibility is extremely reduced, being almost impossible to continue playing.

Figure 10: An example of a visual effect applied to the game's environment depending on the players' stress. Source: (Reynolds 2012)

[82] The game official website can be found at: http://nevermindgame.com/ (accessed on May 15, 2016).
[83] Although we could argue the opposite as Tetris 64, released for the Nintendo 64 game console, as sold with an ear sensor that monitored the players' heart rate and used this data to control the game's speed. This game was only released in Japan, however (Nogueira, et al. 2013, 52).
[84] By mentioning he player's vision, we are in fact talking about the in-game camera which works as the only point-of-view the player has on the game.

A similar example can be seen in **Figure 11**. Depending on the players' stress level, a part of level might start to be filled with milk,[85] as if trying to drown the player. The higher the stress level, the higher the liquid goes, with the opposite also being true.

Figure 11: Another effect of the players' stress level on the game environment.[86]

In terms of visual effects being applied to the players' camera,[87] the reader may observe the appearance of a certain visual static caused by a rise in the stress level in **Figure 12**. This effect is named *stress static* by Erin Elizabeth Reynolds (2012).

Figure 12: An example of *stress static* in the *Nevermind*. Source: (Reynolds 2012)

In the leftmost example, a screenshot of the game environment without any biofeedback mechanics in effect is shown. However, on the right side, the image seen by the players becomes unclear. These *obstacles* are the result of the biofeedback influence, which triggered the mentioned *stress static*.

Another example where biofeedback is used to influence the game's visuals is the experiment performed in *Please Biofeed the Zombies: Enhancing the Gameplay and Display of*

[85] The reason for this choice of term is tied to the game narrative.
[86] This image source (among other screenshots from the game) can be found at: http://www.gizmag.com/nevermind-video-game-biofeedback-stress-levels/29728/pictures (accessed on May 15, 2016).
[87] And by camera, we mean the game camera through which players are able to observe the game world, working as their visual perception.

a Horror Game Using Biofeedback, an article by Andrew Dekker and Erik Champion (2007), where both authors use the players' heartbeat to apply special effects to the their vision.

Every time the players' heartbeat rose above a certain threshold[88] the screen trembled[89] to suggest that the players avatar was out of control(2007, 552). Unfortunately, effects like these should be used with caution as the authors report that the *camera's brusque movement did not please the experiment's users*.

Based on this mechanism, we would like to suggest extending its use to control the players' field-of-view, widening it when serene and narrowing it in stressful situations, In extreme circumstances, it could become so narrow that it would be possible to simulate the tunnel vision condition.[90] Similarly, a mechanism that would reduce the players' visual acuity[91] would also be a good mechanic to convey a sense of despair, as if the player was going to faint at any moment. This is a common situation in horror games, where the players' avatar is desperately trying to escape a pursuing enemy.

Dekker and Champion also use biofeedback to alter or apply different shaders.[92] For example, a reddish filter was applied if the heartbeat was high,[93] to show that the players were getting excited or anxious. If this value kept rising, the color would become more intense until the screen faded completely to red, simulating a *berserk state* (2007, 553). In the opposite case, if players were able to remain calm, the screen would become white or black and white[94] (2007, 553). These changes can be observed in **Figure 13**, with the leftmost screenshot having the default shader applied and with the rightmost presenting the shader resulting from biometric data collected.

The authors also present some conclusions, expressing that the black-and-white filter had a calming effect in players, the red filter did not cause any significant fluctuation in the biometric data collected and the white filter confused players.

[88] With this threshold being 3.8 times the average of the heartbeat measure in the calibration process.
[89] This shaking was possible by applying an effect to the game camera.
[90] Condition where a person completely loses the peripheral vision, giving a sensation similar to looking at the world through a pipe or tunnel. This condition can result, for example, from a high level of stress.
[91] This could be achieved by making the game looking blurry.
[92] If unfamiliar to the concept, a *shader* is a computer program responsible for drawing every pixel, with the appropriate color, in the computer screen according to a set of rules and restrictions. These rules are defined when creating the shader. For a quick explanation of the concept, the reader could watch the following video: https://www.youtube.com/watch?v=TDZMSozKZ20 (accessed on May 16, 2016); and for a detailed clarification, the following link is recommended: https://thebookofshaders.com/01/ (accessed on May 16, 2016).
[93] Two times the average measured in the calibration.
[94] When the heartbeat multiplier was under 0.2 and 0.8 the average measured in the calibration phase, respectively.

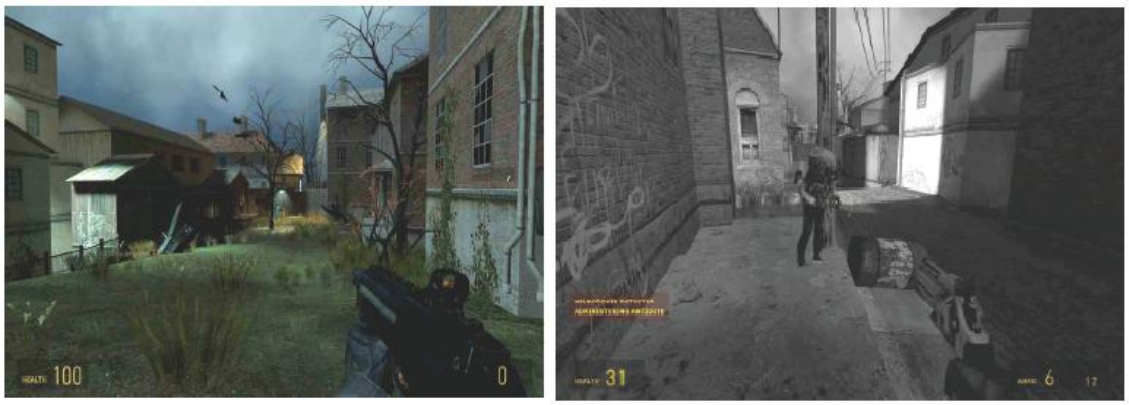


Figure 13: Players' affective state controlling the shaders applied in the game. Source: (Dekker and Champion 2007)

Besides the visual component, sound is also considered a part of the game aesthetics, with Dekker and Champion also being an example of this fact, as they had their game background music adjusted dynamically based on the players' sweating.[95] We argue that this same mechanism could be extended to feature affective states, valence, arousal and (perhaps) other biometric values as a source of this adjustment.

In addition, to avoid players staying in a calm state of mind for too long,[96] a random unsettling sound clip (for example, a scream) was played with the intent of scaring players. The sound clip of a beating heart was also played, with its frequency matching the players' own heart rate, which the *participants claimed that it made them feel more involved in the game*.

Based on the previous example, we would like to suggest the use of biofeedback to apply effects in the background music or sound effects played during the game. Just as Dekker and Champion changed the heart beating sound file play rate to match the players, we advocate that different effects (such as low or high-pass filters, equalizers) could also be applied through a similar method.

Summarizing, it was possible to determine that biofeedback can be used to:

- Add, change and remove certain elements (game assets) contained in the game environment;
- Decide which textures to apply to game assets;
- Apply or remove visual effects in the game environment (mist);
- Apply or remove visual effects in the player's camera (visual noise, movement, change field-of-view, change visual acuity);
- Apply, change or remove *shaders* (transparency, colour filters representing the player's affective states);
- Control the volume of the background music being played;

[95] In other words, based on the players galvanic skin response.
[96] In other words, to ensure that the heart rate and galvanic skin conductance were not inferior to the values captured during calibration.

- Change the background music based on the player's affective state;
- Trigger sound effects in order to arouse, reassure or calm down the player;
- Trigger sound effects that inform the player of his/her affective state (a sound clip of a beating heart altered to have the same heart rate as the player);
- Change the volume of the sound effects;
- Apply effects (filters, equalizers, among others) to sound.

3.1.2 Dynamically adapt the players' actions and performance

One more time, Dekker and Champion's (2007) experiment is used as an example, with the heartbeat and the galvanic skin response being used to influence the game's mechanics and, consequently, the players' actions and performance.

One example is the *stealth* mechanic implemented which is directly related to the mentioned biometric values. When these values went below a certain threshold[97] the players would become invisible and undetectable by the enemies. The damage inflicted by the players' avatar was also dependent on these. The *running* mechanic as the players' avatar movement directly mapped to the their heartbeat.

Other mechanic worth mentioning is the *Bullet Time*[98] effect: "[t]his effect changes the density and gravity of the environment, to emulate the effect of the avatar being faster than other characters" (2007, 552). In this specific case, the component suffering the change is not the game mechanic itself but game engine, who has its configurations altered to achieve the intended effect.

Dekker and Champion's also registered some players' feedback, with one of suggesting that the avatar's life could also be tied to the heart rate. Using this as starting point, we could also suggest that players' life regeneration mechanic, if one exists, could also be conditioned by it, easing or hindering the recovery depending on its frequency.

Other interesting, and uncommon, mechanics found is the *diving* and *possessing* mechanics proposed by Gonçalo Silva (2014).

Starting with the *diving* mechanic, the author tried to replicate the human behaviour while diving, where one must avoid breathing underwater, forcing one to return to the surface to renew the oxygen level, saving oneself from drowning. Based on this, Silva mapped the respiration sensor used on his experiment to the diving mechanic. This way the players' should

[97] Half the values measured in the calibration stage.

[98] *Bullet Time* is a visual effect used in several fields, in which cinema and video-games are include, resulting in a effect similar *slow motion*, although the process used to achieved both are different. In cinema, *slow motion* tends to reproduce the captured frames at slower rate. *Bullet Time* are several cameras capturing the same scene from different angles and then playing a frame of each in a defined sequence.
In general terms, *Bullet Time* can be seen as a slowing down time to show imperceptible and unobservable details, such as showing the movement and trail of a bullet after being shot (whence its name).
An example of this effect can be found at the following link: https://www.youtube.com/watch?v=bKEcElcTUMk (accessed on February 29, 2016).

avoid breathing while their avatar is swimming underwater otherwise a penalty would be applied to the avatar's health.

3.1.3 Complement the game's controller as a form of input

Based on the examples presented in the previous section, it becomes clear that mapping certain mechanics directly to biometric inputs seems to be a common and accepted pattern in games with biofeedback. This way, biofeedback naturally becomes a complement to the game controller. By mapping some biosignals to certain game mechanics, players are able to trigger these through the use of their body.

Rego (2014), with the objective of assimilating players into the game itself, developed a tool that was able to receive, process and map a set of physiological data to a set of controls (keyboard inputs) that would then be sent to the video game. These inputs would trigger certain mechanics such as attacking or defending, among others. To perform his experiment, he used the game *Dark Souls* (2012).[99]

In this game, the players' avatar is able to perform, among many others, four specific actions: attacking, defending, running and healing. The avatar carries its shield on the left hand and a sword on its right. In order to mimic this setup, Rego used three EMG sensors, one in the left bicep, other in the right bicep and a last one in one leg. When the values captured by the sensors went above a certain threshold, they would trigger, respectively the action to defend,[100] attack[101] and run.[102] A respiration sensor was also used to heal the player.[103]

One other game mechanics developed by Silva[104] (2014) is the *possessing* mechanic. This allowed players to take control of an enemy body, forcing it to do their bidding or commit suicide. This was achieved by using two mechanisms. In the first "(...) players Possess enemies by blowing hot air on the TEMP[erature] sensor, which mimics the player's soul leaving the body as it enters the possession's target", while in the second the player performed a gesture with their hands (one gesture for possessing and releasing the target and another for suicide) while blowing hot air to the temperature sensor.

Based on the two previous examples, we can conclude that typically biofeedback complements game controllers because they permit mapping one's actions to the actions of avatar in the game environment. Game mechanics are created with these actions on mind,

[99] Dark Souls was published in different platforms at different times. The version used in Rego experiment as the PC version.
[100] The contraction of the left bicep could be seen as the players pulling their shield up to defend an attack.
[101] The contraction of the right bicep could be seen as the movements and contractions the muscles on the right hand need to perform to land a blow on the enemy.
[102] The contraction of the leg could be mimic the action of running.
[103] The act of breathing out could be seen as a sign of relief by the players after recovering from injury.
[104] In the previous section, we presented the diving mechanic developed by the same.

implemented into a game and triggered just as one would if performing them in the physical world.

3.1.4 Control the player's avatar behaviour, precision and performance

In a game, players are represented by an avatar through which they are able to shape or interact with the game world. Optimally, this avatar should be responsive, interpreting the player's input as accurately as possible, in order to perform their intentions. However, the avatar could, in certain situations, refuse to acknowledge the players' input or react slowly and inaccurately, as if it had its own affective states.

For example, in the game *Heavy Rain* (2010), the possibility of running is context dependent. Inside one of the playable character's home, he simply walks around, portraying how typically people move inside their houses.[105] However, in certain conditions, such as those that demand quick and effective actions, where running does not seem like an unreasonable attitude from a human point of view, this action is possible.[106] In *Fallout 4* (2015), the players' avatar is not allowed to run when the weight of the items they are carrying exceeds the maximum capacity that the avatar is able to carry or when they have hurt their legs.

If however, the players' affective state could be transposed to their avatar, the avatar could have a matching behavior. If the player is nervous, the avatar is also nervous, making it possible to run inside is home. Are the players desperately trying to run from the monster pursuing them? In that case, so is the avatar, who muster enough adrenaline to run and carry the excess loot even when hurt. Of course, the opposite should also happen. If the players are scared of facing an enemy why should their avatar be willing to? Perhaps having a downgraded performance when fighting it would be the correct outcome (although not the best outcome in the players' point-of-view).

In the video game *Atelier Shallie* (2015), one of the game's core mechanics is the main character motivation. As the game advances, a set of tasks will pop up, representing character wishes. Successfully completing these makes the character (which is also the players' avatar) more motivated or, if the maximum limit has been achieved, keep her in high morale. However, as time goes by, morale starts decreasing. Therefore neglecting these tasks will result in a depressed character, having the players to manage how much time they dedicate advancing the story and how much is spent in fulfilling these. The morale (or mood) is presented in the form of a bar, with a smile above it, as seen in **Figure 14**, which the player can check anytime.

[105] The following video contains some footage where the player explores the main character's house. This exploration is done slowly as the player is unable to make the character move faster: https://youtu.be/UVSVh5gtREc?t=8m50s (accessed on May 17, 2016).

[106] The following footage presents a situation where the players are now able to make the character they are controlling move faster, almost running, contrasting with the previous example: https://youtu.be/ezZWqN9htC4?t=3m37s (accessed on May 17, 2016). In fact, the player may now be unable to make the avatar walk due to the stressful situation he is facing.

Figure 14: Screenshots of the main character's motivation bar in *Atelier Shallie*.

The motivation affects many other mechanics of the game. For example, when gathering materials (a essential action in the game), a highly motivated character will collect many more material (five materials per search) than when on a neutral[107] state (three materials per search) or when depressed (one material per search). Also, a motivated character will move much faster than a depressed character, which will have a very slow pace. In other words, the same action or mechanic is slightly altered based on the affective state of the avatar. If we imagine that bar has the player motivation or affective state, then we would have the avatar performance matching the players'.

Summarizing, the biometric input could be used to:

- Define the avatar's current mood, changing his behaviour accordingly;
- Control the precision and performance with which the avatar's responds to the player's input (highly accurate, slowly, incorrectly or unresponsive).

3.1.5 Control the elements that act in opposition or a neutral to the player

Biofeedback could be used to control the opposing elements to the players progress. These elements commonly take the form of enemies, traps or other gimmicks that challenge and oppose the player.

In Dekker and Champion's experiment (2007) the enemy generation is tied to the players' heart rate: if the player is calm,[108] a common enemy is created to counter it. However, if they remain untroubled,[109] a powerful enemy (also known as the *Boss enemy*) is placed in game. Similarly, Torres uses the players' arousal to control the probability of a creature (the enemy from which the players must run) appearing on the level: if arousal is high, the chances of appearance are low, with opposite also being valid (2013, 41).

Also, Dekker and Champion's *stealth* mechanic, described in section **3.1.2 Dynamically adapt the players' actions and performance**, could also be included in this section if, in some way, it changes the enemies behaviour[110] to ignore the players when their biometric values are bellow the defined threshold.

[107] Neutral state means that the character motivation is about 50%.
[108] Whenever the heart rate multiplier is lower than 0.8 times the value measured during calibration, the player is considered in a tranquil state.
[109] If the heart rate multiplier becomes lower than 0.4 times the value measured during calibration.
[110] The enemy artificial intelligence.

Just as the opposing elements can be controlled, the neutral elements, such as non-playable characters (NPC), can be controlled the same way.

3.1.6 Control the game's difficulty

In the previous section, when referring to the opposing elements, we were clearly dealing with challenge and difficulty offered in a game. As such, we could also say that biometric input that change the game's difficulty could be included in previous group. Commonly, decreasing the game difficulty would mean giving more clues to the players on how to solve the current puzzle, weaken the enemy performance, improve the player performance, remove certain type of obstacles, adapt some game mechanics (such as the speed the platforms move), among other steps. Many of these examples would fit in other sections[111] and, as such, we thought it would be adequate to create a separate group to approach this subject.

Several games already offer the possibility of setting the difficulty at start, remaining constant during the remainder, while other offer the chance of changing it midgame. This change can happen automatically,[112] at specific moments in time or when the players decides. As shown before, *Super Mario 3D World* (2013) offers a *power-up* to ease the challenge when the players lose five times in a row, while *Bravely Default* (2013) offers the possibility of changing the difficulty on the fly by accessing the *config* menu, as shown in **Figure 15**.

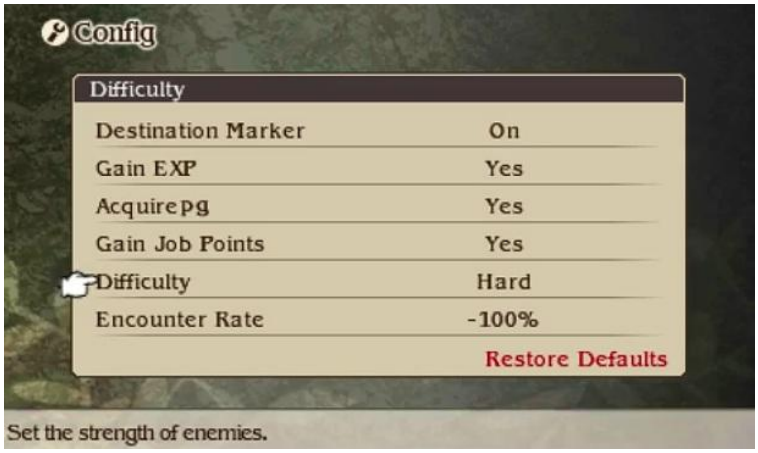


Figure 15: Changing difficulty in *Bravely Default* (2013).

One way of applying biofeedback in this category would be to develop a system that allows switching the game difficulty at any moment and map it, for example, to the players'

[111] Giving more clues on how to solve a puzzle could be seen as altering the NPCs dialogues (section **3.1.7**); weakening the enemy performance or removing certain type of obstacles could be seen as controlling the opposing elements (section **3.1.5**); and improving the player performance could be seen as adapting the player's actions and performance (section **3.1.2)** or improving the behaviour and performance of the avatar (section **3.1.4**).

[112] With the help of profiling, with the game *Left 4 Dead* (2008), described in section **2.5 Profiling**, being one example.

arousal. This way, if this value remained excessively high during a certain period of time (which might suggest that the player is under *stress*) the difficulty would be reduced automatically and, if the opposite happened, the difficulty would be incremented. This is just an example, with other mappings being possible: Ambinder (2011) used the players' skin response during his experiment.[113]

3.1.7 Change or adapt the game's narrative

The players' biometric data could also be used to change or influence story or narrative of a game by altering the dialogues to fit the players' affective state, by leading the players into a certain mood or by completely altering the narrative, sending then into a new story branch.

The first option could be achieved by, for example, changing the NPCs' dialogue to the affective state of the players. This way, different affective states would have different dialogues matching them, creating a kind of branching dialogue but where the choice is made automatically by the system based on the biofeedback. **Table 5** presents variances on the dialogue depending on the player's affective state.

Table 5: Variations on dialogue depending on the player's affective state

Affective State[114]	Dialogue
Neutral	"Be welcome to (somewhere), the land of shinning dreams. Our fresh breeze is know for inspiring brilliant ideas in everyone, being them poets, scientists, artists or con... *cof* Other kind of artists!"
Happy	"Ah! A radiant face! Full of dreams and hope, nay? You are just at the right place! Welcome to (somewhere)!"
Calm	"Be welcome to (somewhere), the land of shinning dreams. Our fresh breeze is know for inspiring brilliant ideas in everyone, being them poets, scientists, artists or con... *cof* Other kind of artists!"
Tense	"H-H-hello... Be wel... Welcome. *gulp*"
Sad	"Egads, such a depressing visage you bear. Cometh, cometh! There is nothing better than our breeze to heal disease that may hail thee, body or heart!"

[113] This experiment has also been described in section **2.5 Profiling** where Ambinder fed *Left 4 Dead* 2's artificial agent with the players' skin response in order to determine a more accurate arousal value.
[114] The affective states presented in Table 5 are based on the simplification made by Zagalo of Russell's circumplex model of affect, as show previously on section **2.2.1**, **Figure 3**.

The neutral state can be seen as the solution for when the biofeedback is unavailable, an error occurred or the system is unable to determine the current player's affective state. Also, the same dialogue line could be shared between affective states, as it happens with *calm* and *neutral*.

As seen, the dialogue has been adapted to the players' current affective state. When the system detects that the players are *happy*, it makes the non-playable character interact with players in amore effusive way, commenting their good mood. On the other hand, when the conclusion is that the player is feeling unhappy, the NPC tries to cheer the players. Now, if the players are *tense*, the character responds warily, omitting much of the information it had to give.

One important point to highlight is that there is no correct or wrong affective state. In this specific example, as a way to get the most of information, the players should try to avoid being *tense* and if they were *sad* they would end up not knowing the location's name. However, imagine a suspect interrogation where the players are the inquisitor. Affective states such as *calm* could lead the subject to hide information or lie. A *tense* state, on the other hand, could lead to a fortuitous result. It is up to the game designer to choose how to use this mechanism: it could use it as a game mechanic for unlocking information or could use it as a simple way to adapt the story the players' mood.

Another option could be to lead the players into a certain mood. One example that tries to use a similar mechanic, although without the biofeedback, is the game *Super Metroid* (1994). Right at the beginning of the game, the players are forced into fight with one powerful enemy which they will have barely any chance to defeat, seeing as this is their first battle in the game. The players will be required to do their utmost to defeat it, possibly generating stress ("when does the enemy die?") and/or frustration ("how can I fight such a powerful enemy in such a limited space with such a weak weapon?"). However, right as the players are about to die, the enemy flies away, leaving the players to live another day.[115]

In truth, this was scripted event, leaving the sensation that game designers were trying to, artificially, create a tense and challenging mood and then a sense of relief (only to be followed, seconds later, by another stressing situation). All this is a illusion, as the players could remain completely still until the health threshold was reached and the enemy left. By using biofeedback, we could measure the players' arousal or affective state until these value reached the intended value, and then making the powerful opponent leave. This way, situations such as the players remaining still could be avoided, as their mood would not swing.

Of course, there is always the possibility that the player could handle the challenge and manage to defeat the enemy. Nevertheless, looking at the stage and enemy abilities, we believe the game is rigged[116] in favour of the enemy and scripted event. If the players managed to win,

[115] This specific event can be observed at the following video: https://youtu.be/byqCZJwCt3I?t=3m28s (accessed June 23, 2016). This event takes place between 3m28s and 4m10s, with remainder of the video being unrelated.

[116] The stage is relatively small, the enemy occupies a great part of the level and is able to move and reach any area rapidly, its abilities involve shooting fire that spreads thorough the area (forcing the player to jump, which will surely

it would indeed have been a real challenge with a well deserved sense of accomplishment and relief.

The third way of using the players' biofeedback is changing the game narrative. This change does not have to be a drastic change, it could simply take the form of choosing one of the games many branching stories automatically. Depending on the affective state, arousal or valence of the player, the adequate story branch would be selected. Are the players in a high state of arousal? Than perhaps a narrative branch that offers more action would be the correct choice. Did the last story sequence or cutscene[117] leave the players in sorrowful tone? Then perhaps a slow paced narrative filled with happy events would be the way to go; or was the intention of the game designer making the players even more depressed?

Summarizing, biofeedback could be used to:

- Alter dialogues presented in game (NPCs dialogue);
- Influence players into a specific state of mind or affective state that matches the intended mood of the game designer;
- Automatically select which branch narrative better fits the players;

3.1.8 Manipulate the dramatic tension in a game

This section will suggest adding biofeedback to LeBlanc's (2005) mechanisms[118] used for manipulating the dramatic tension in a game.

The first suggesting is adding biofeedback to the positive and negative feedback systems. This way, using as an example a racing game between two players, whenever one player's *stress*[119] level became low,[120] the negative feedback system would be activated, closing the distance between both players, increasing the dramatic tension. However, if the players' stress level became too high or if the game became stagnant (the players' stress level went back to being low), the positive feedback system could be activated correct this problem.

One other suggesting is altering the *Fog of War* mechanic to include biofeedback. In the game *League of Legends* (2009),[121] the players are able to hide in specific spot in the map

result in colliding with the enemy) or spinning its tail. Furthermore, this is the first battle players face and there were no targets, previously, that the players could use to understand and train the shooting mechanic.

[117] Cutscenes are "non-interactive sequences used by many games to provide backstory, advance the plot, or illustrate objectives for the player to complete". Source: http://www.giantbomb.com/cutscene/3015-22/ (accessed June 23, 2016).

[118] These mechanism have already been detailed in section **2.6 Dramatic tension in games**.

[119] The stress could be determined by the arousal value or by the affective the players are currently in.

[120] This low value could result from the players finding themselves in first place, confidant that they are able to keep this position, or from being in the last position, having already given up the competition, awaiting their defeat.

[121] *Leage of Legends* is a third person multiplayer online battle arena where players compete between teams to fulfill a victory condition, which is typically destroying the opposite team stronghold or keep control of certain strategic point for a period of time. To achieve this, the players are given an avatar, from a set of pre-made characters, which

making them undetectable by the opposing team while they remain there.[122] Furthermore, while the players of one team are outside of the opposing team field-of-view, their positions and actions cannot be observed (due to the *Fog of War*). Now, we ask the reader to picture the previous mechanics having biofeedback influencing it, where the players could only remain undetected if their heartbeat and breathing were below a certain threshold. When disrespected, the player's position would be revealed despite being in a hiding spot. Extending this mechanic to the *Fog of War*, if the previous values were disrespected, the players' actions and position would be revealed to every player thus hindering the player's tactics. One other interesting use is doing the opposite: whenever the players were able to keep their biometrics values under control, they would become undetectable to the adversaries wherever they were. This previous mechanic would have an interesting application in a *stealth* video game.

Regarding LeBlanc's *ticking clock*, it would also be interesting to apply biofeedback to it, using the players' unrest to influence its speed. In the game *Keep Talking and Nobody Explodes*[123] (2015) players must disarm a bomb within *a time limit*, with this *ticking clock* taking the form of a red electronic timer. Imagine, now that the biometric data of all the players is collected and use to control the bomb timer, influencing the rate at which the timer goes down. These could be used to provide to a more dramatic or relaxed experience, with the timer decreasing slower or faster depending on the players' arousal or affective state. The players' eye movements could also be used to determine if they were looking at the timer, pausing the effect if the gaze as centered or around it, fooling the players into believing that no mechanism was at work..

Summarizing, biofeedback could be used to control the dramatic tension mechanisms, namely:

- Activate and deactivate the positive and negative feedback systems;
- Control the effects of the *Fog of War*;
- Influence the *ticking clock* functionalities, such as the rate it decreases per second.

3.1.9 Generation the game level layout

Much to the author surprise, biofeedback could also be used has an input or a way to control the mechanism for generating a level layout. This idea came from the experiment performed by Torres (2013), where he tinkered with layout generation of the game *VANISH*

---

they control to do their bidding. The players can move, attack opposite team members, cast their avatar's special abilities, shop for upgrades, hide in specific spots on the map, among other actions.
The game's main webpage can found at: www.leagueoflegends.com/ (accessed June 13, 2016).

[122] A pseudo *Fog of War* mechanic.

[123] In this game one player must disarm a bomb, within a time limit, with a help of a friend or friends, but with a twist: only the main player has access and can see the bomb. The remaining players, however, possess the manual with the steps necessary to disarm it, so players must communicate between each other, correctly and fast enough, to complete this challenge. The disarming process involves solving a set of puzzles (cutting a specific wire, pressing buttons in a specific order, among others).
More information on this game can be found at: http://www.keeptalkinggame.com/ (accessed June 13, 2016).

(2013)[124] to react to the players biological behaviour. In this game players must reach the exit of dungeon where they are trapped, without being captured by the monster that stalks them, while the layout is being changed in real time. The layout generation can be observed in **Figure 16**.

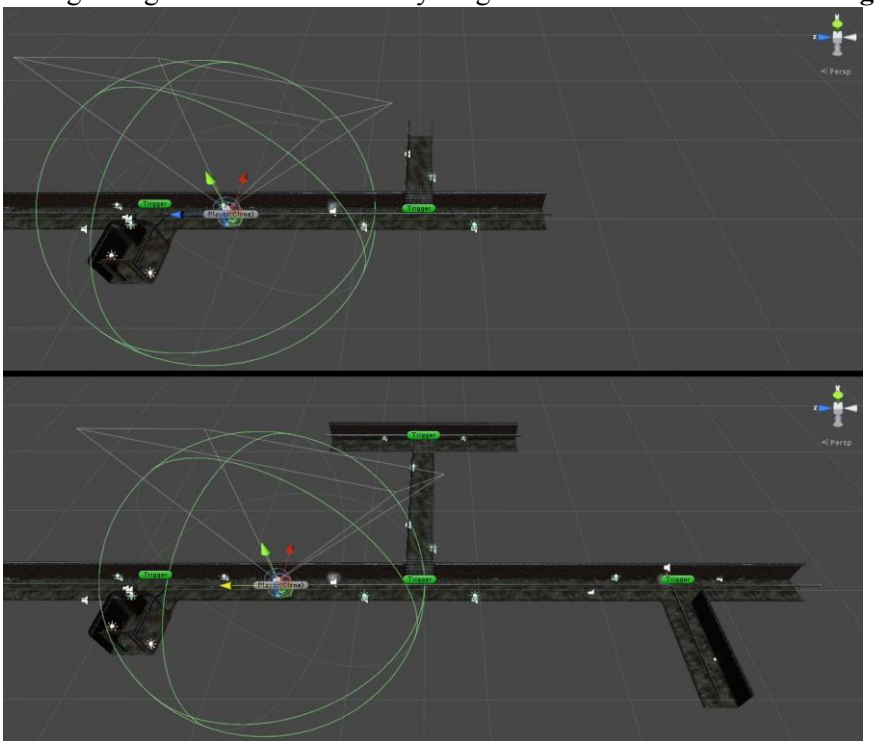


Figure 16: The generation, in real-time, of the level layout in *VANISH*. Source: (Torres 2013)

Here the players' valence directly affects the probability of a key room[125] or an exit room[126] appearing: the lower the valence (the negative the affective state the player is in) the higher the chances of these appearing, with the opposite also being true. When players are being pursued by the enemy, valence also influences the chances of an evasion tunnel[127] materializing.

All these nine distinct groups will be used as inspiration when designing the game mechanics to be used in the case study presented in this work. It is also important to remind the reader that there be many more uses for biometric feedback in video games. These groups are not definite, with the possibility of being expanded.

[124] *VANISH* (2013) is survival horror game where events and map sections (the layout of the level) are random and generated procedurally (Torres 2013, 24). The game website can be found at: http://www.vanishgame.com/ (accessed June 23, 2016), where the game can be freely downloaded.
[125] A room which the player must visit to collect an item. When two of these items are gathered (when two key rooms are visited) the exit room starts appearing.
[126] The exit room is the room where players first started and the place they must reach to complete the game, after having visited the two key rooms.
[127] A room where players are able to hide from their pursuer.

The next section will then deal with the conceptualization of the framework, which will support the integration of biofeedback in games that wish to use it and, consequently, make the previous game mechanics implementable.

3.2 General description of the framework

According to Russell (1980), two elements are needed to determine one's affective state: valence and arousal.[128] However, to determine these, an assortment of biosignals must be collected from the player and analysed. This way, a certain chain of dependences starts to form: affective states depend on valence and arousal, and these are dependent on the biological data, as shown in **Figure 17**.

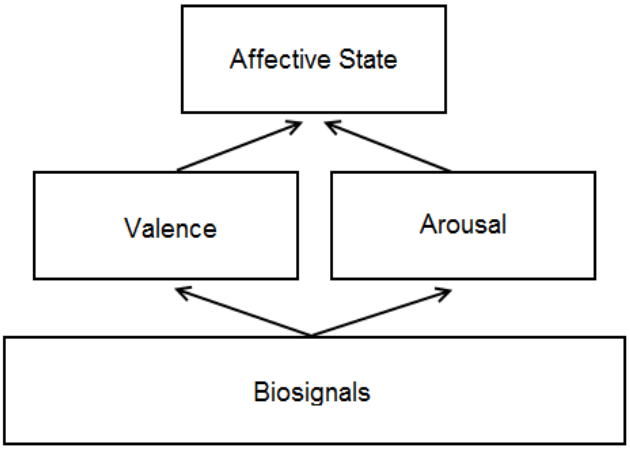


Figure 17: A chain of dependence between the four elements in the framework.

Each biosignal must be captured and processed by a sensor, transforming the raw signal into a value understandable by the intended receiver. Therefore we need to reformulate the previous diagram to better reflect this, as show in **Figure 18**.

[128] More information on Russell's circumplex model of affect and his two *neurophysiological systems* can be found on section **2.2.1**.

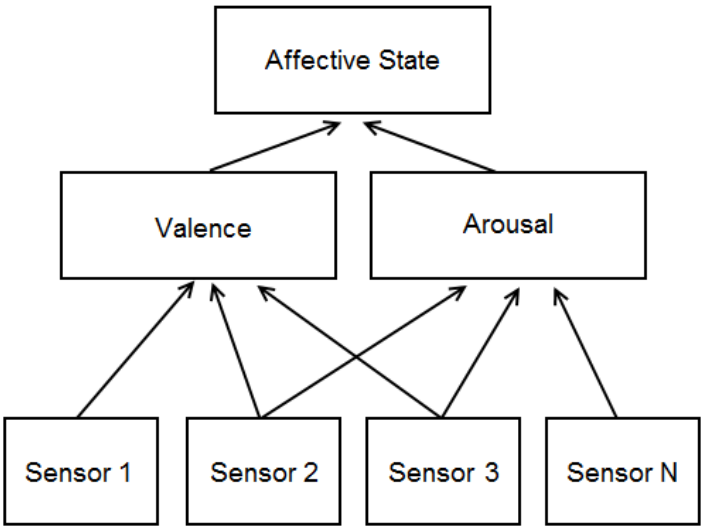


Figure 18: Sensors as the source of biosignals in the framework.[129]

Two important points must be highlighted: the framework should be able to accept a varying number of sensors; and not all sensors contribute equally to determining player's valence and arousal: some sensor contribute to determine arousal (sensor 1 is an example), while others valence (sensor N), because the biosignals they capture do not confer the necessary data.

However, one thing that **Figure 18** does not portray is the necessary complexity to transform the data processed and given by the sensors into valence and arousal; and these two values into an affective state. At some point, during the process, two interfaces must exist: one responsible for collecting all the information given from the sensors, determining valence and arousal, and another that is able to receive the computed values, search for the combination of both in the circumplex model of affect and determine the affective state. Once more, the previous diagram is updated in **Figure 19**.

[129] Sensor N, present in the diagram, is meant to show that the framework can contain a varying number of sensors.

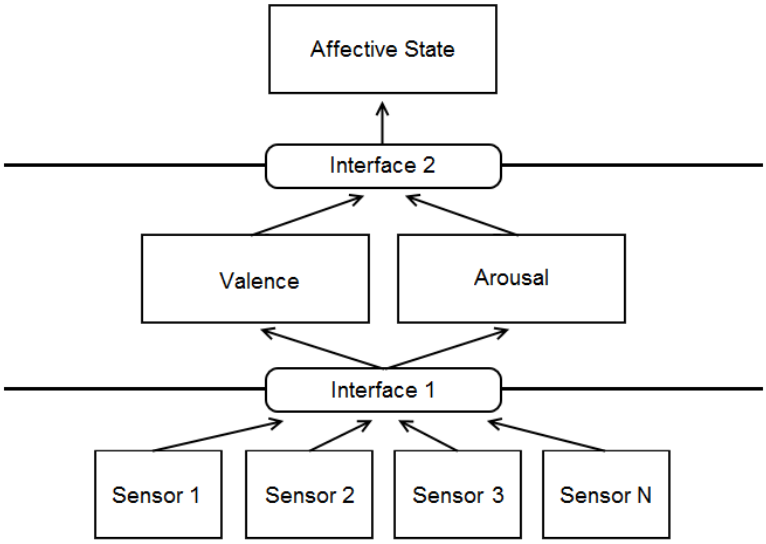


Figure 19: The two interfaces present in the framework.

Based on this, a hierarchical relationship emerges between the components: the sensors, at the bottom, responsible for collecting the biosignals from the player, serve as the base for all the elements above it, being the source for the valence and arousal, which are the result of the calculations performed by Interface 1[130] on the sensor data, and the affective state is the result of the operations performed by Interface 2 using the valence and arousal . This hierarchy can be better observed in **Figure 20**.

[130] This process will vary from author to author. We present Nogueira's example in section **2.4**, highlighting the PIERS sub-system.
Regarding sensor calibration, a typical step in biofeedback games where the sensor values received are adapted to better match each player, we considered this topic to be outside of this dissertation's scope. However, we believe that this step may happen in three distinct moments: 1) before the biometric data is received by the framework, being performed by an external component, 2) while the biometric data is being processed inside the framework, possibly being performed by Interface 1 or a complementary interface, or 3) at some point during the game, with this step being performed by the game as well. However, to determine which solution is feasible and better suited, a detailed study must first be carried out.

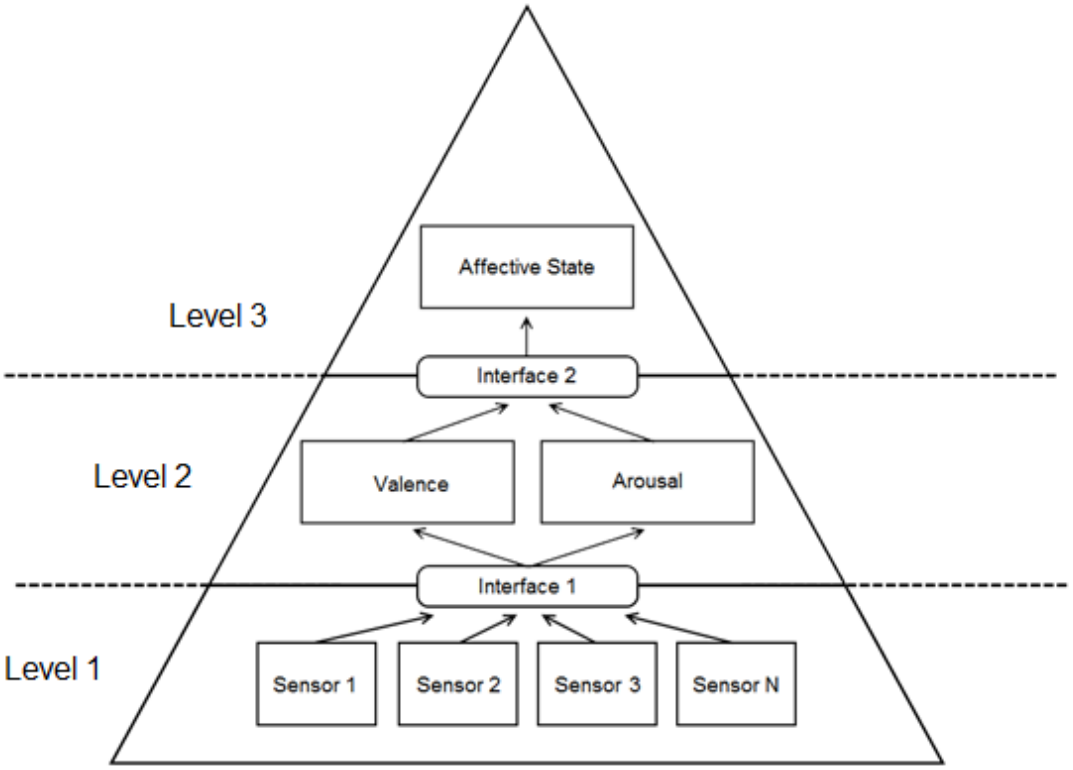


Figure 20: Framework's hierarchy.

With this in mind, the first component of the framework can also by divided in three levels, using interfaces 1 and 2 as diving lines, with each handling a particular type of data:

- **Level 1**, at the bottom of the pyramid, where the non-processed[131] signals provided by the sensors are dealt with. This level can be seen as the lower-level of the framework as we are working with raw data;
- **Level 2**, in the middle of the pyramid, where arousal and valence values are dealt with. This can be seen as the mid-level because these values result from the processing of the lower-level data and contain a higher level of abstraction than the previous level;
- **Level 3**, at the top of the pyramid, where affective states are dealt with. These are high-level abstractions of the values collected on the previous level.

All the data collected during this process, from the lower to the higher level, is sent to the game, separated by level. This way, all the information collected in the process is made available to the designers and developers allowing them to choose on which level to focus (they could focus on all levels) depending on their needs and knowledge. How is this information used, however, is left to the designers of the game.

[131] The values provided by the sensors are indeed processed, they have been converted from their original value into one understandable by receiver. However, from the receiver point of view, these values are seen as raw, non-processed values of the biosignal.

Figure 21 shows the updated diagram of the framework, in which is possible to identify two elements: the left one (the pyramid), responsible for the collection and processing of the data, and the right one, which is the game. It is also possible to see the position of the players in the framework process: they are the ones who provide the biometric data that feeds the framework and are the observers of its changes in the game.

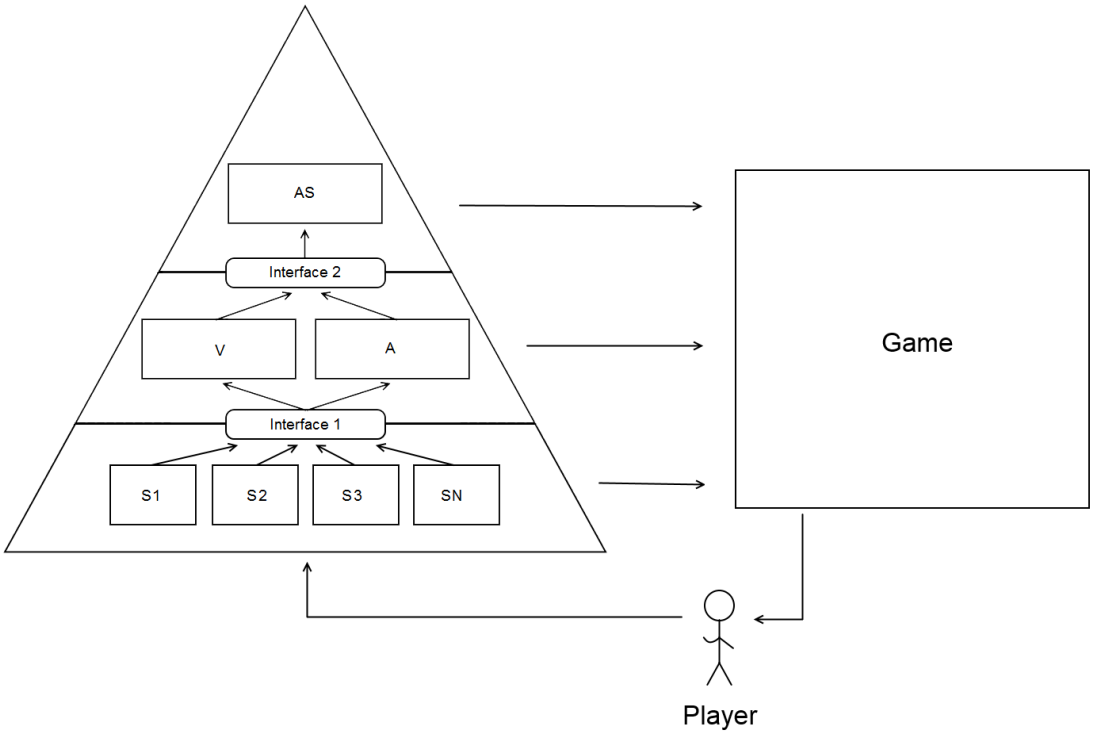


Figure 21: Three essential components of the framework.

When studying the previous diagram we found an opportunity to improve the current framework in order to ease the design and implementation of the biofeedback in video games. We thought of adding one more interface, a new component located between the data collection component and the game, that would receive the collected information and translate it into an input understandable by the game. This was accomplished by means of a rules system[132] that the game designers could set, beforehand, through the use of a simplified notation language. **Figure 22** updates the previous diagram and represents this framework final version.

[132] We will address this issue with more detail in the following section.

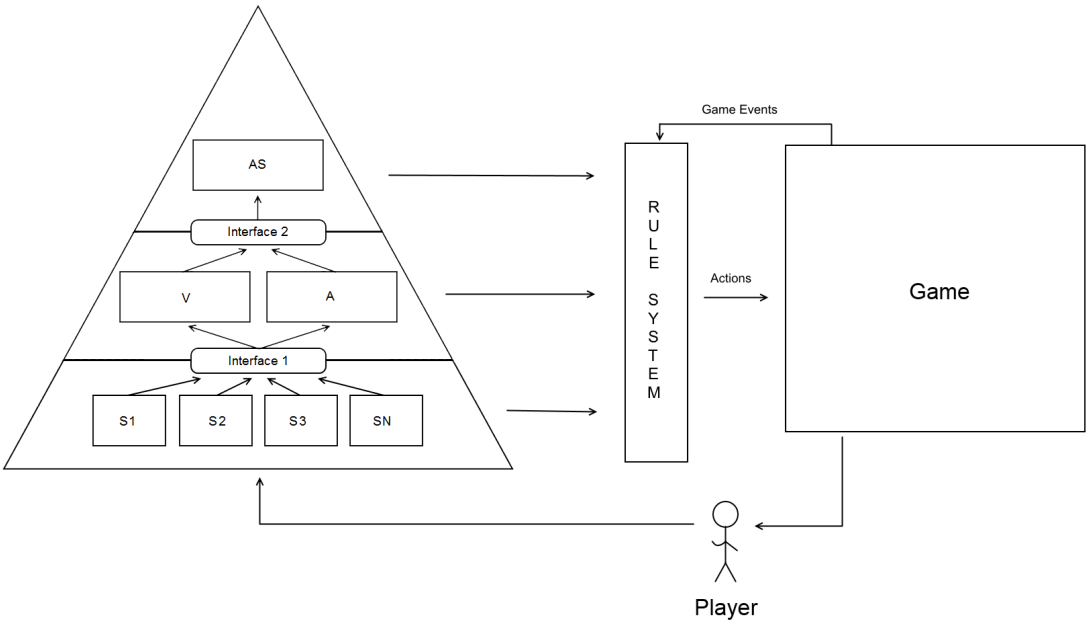


Figure 22: A general view of the framework featuring the rules system.

3.2.1 The rules system

The rules system, as previously mentioned, is a component responsible for receiving the biometric information and translate it into an input understandable by the game. From now on, we will call these inputs: *actions*, an instruction, value or similar that will be received by the game, leading it to alter (or not) some part of its behaviour. This process can be observed in **Figure 23**.

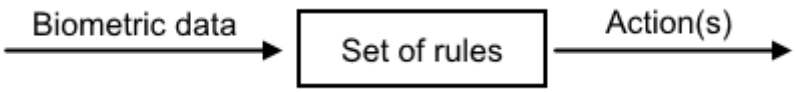


Figure 23: The translation process.

In order for this translation process to occur, the system will make use of the rules defined, beforehand, by the game designer. A rule can be seen as a condition that compares one *variable*[133] to a value, state or another *variable* specified by the game designers.[134] A rule must also identify an output: the said *action* that the system will trigger to affect the game.

An example of a rule is: "when *arousal* is *higher* than *0.8* the *avatar's movement speed will change to 5."* In this example, *arousal* is the variable provided by the system; *is higher* is the comparison the rule will make; *0.8* is the value specified by the game designer and *avatar's*

[133] A variable is the affective state, valence, arousal or value captured by one specific sensor in the previous component (the collector and processor component) and given to the rule system.
[134] The rule system should provide the game designers with all the variables available in the framework (depending on the implementation, this might not be possible). However, we strongly support this, as the less variables the game designer have, the more restricted they will be when designing rules.

movement speed will change to 5 constitutes the action. Based on this, whenever the arousal value received by the rules system is higher than 0.8, the system will warn the game to change the avatar's movement speed to 5. A summary of a rule structure is present in **Table 6**.

Table 6: The structure of a rule.

Variable	Comparison	Value to compare to	Action
Arousal	*Higher than*	0.8	Change avatar's movement speed to 5

It is also be possible to concatenate several conditions in one rule. For example: if *valence* is *higher or equal* to *0.7 and arousal* is *lower* than *0* then *the avatar's face texture should be changed to 'avatar_face_calm'*.

Based on this, the previous diagram can be expanded to match one presented in **Figure 24**.

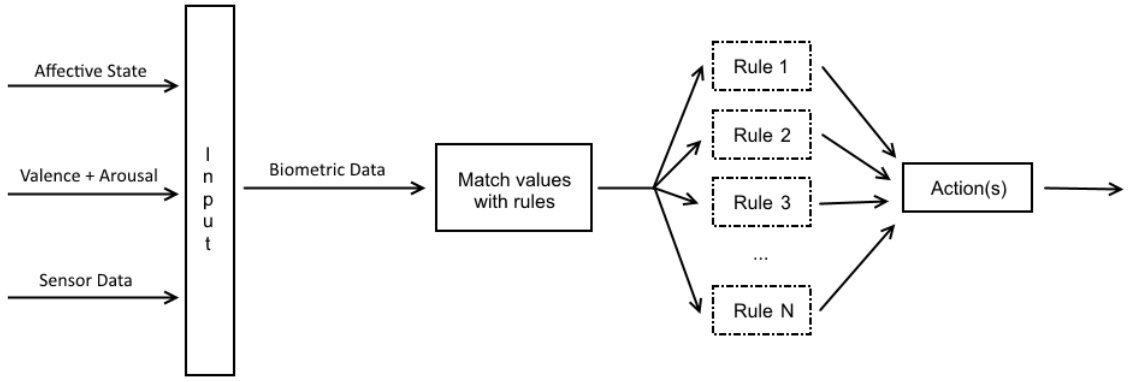


Figure 24: A more in-depth view of the rules system.

Two other variables that are available to in the rules system and that are not provided by the biometric data that it receives are *time* and *game events*.

Time is a variable containing a certain amount of time. However, we find that this variable should be divided into two: *absolute time* and *relative time*. The former contains the amount of time that went by since the game begun. The latter works as a stopwatch, resettable by the game designer whenever necessary, and containing the amount of time that passed from the moment it was started/restarted.

Game events are messages or inputs, sent from the game, that the system is able to use capture and use as another variable. An example can be: when the *game event* received is *player in boss battle* and *affective state* is *tense* then the *boss enemy behaviour* should be changed to *prudent*. Optionally, these *events* could also double as an input, a message understandable by the system, changing (or not) its behaviour or functionalities. These events can be, for example, messages for stopping the rules system in situations where it is not needed or to restart the relative time counter.

To reflect this, we present **Figure 25** with an updated diagram of the rules system.

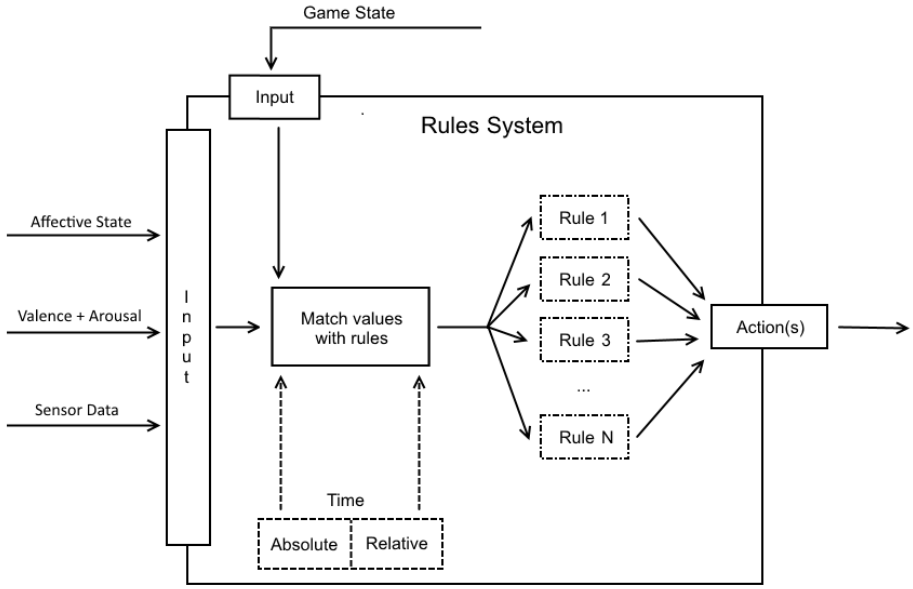


Figure 25: Time and game state variables added to the rules system.

One other functionality that a system like this must possess is being able to store samples, if the game designer deems it necessary. A *sample* can be seen as a *photograph* of all the variables values at a single moment, containing the affective state, valence, arousal, all sensors' data, events received (if any) and time (absolute and relative). One or more samples are able to be used in rules. For example: if the *current affective state* is *different* from the *previous affective state* then (do something).

With this last functionality, the final version of the rules system diagram can be seen in **Figure 26**.

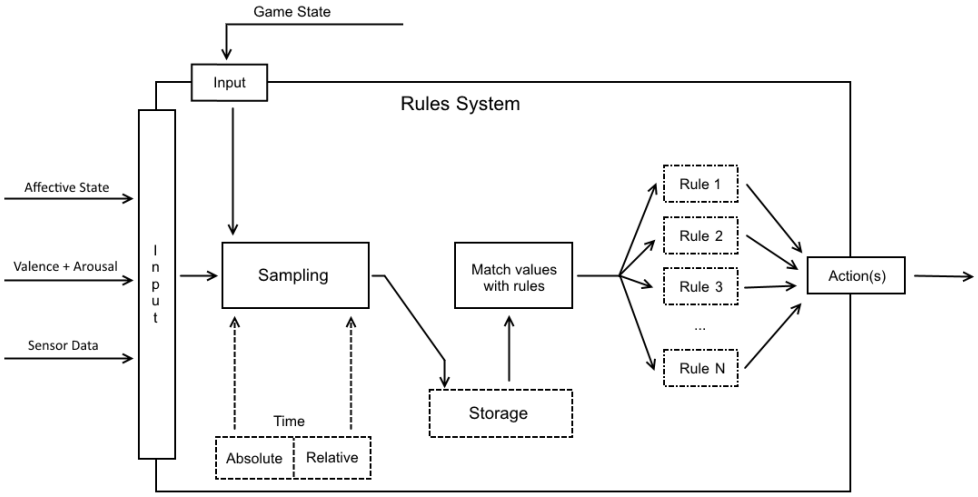


Figure 26: Rules system final diagram.

Depending on the number of samples stored and their use, the rules system structure can be altered to better adapt to the end in mind. The following section will present three distinct possibilities.

3.2.2 Three ways to handle input data history

Depending on the quantity of information (the number of samples) the rules system has to store, three different situations were identified and a solution was developed for each. These solutions vary on how the storage component of the system is implemented.

3.2.2.1 Solution 1: Not storing samples

The first solution consists on the system using only the current and up-to-date sample, whence no storage would be needed. This solution would make the framework closely match one of its previous iteration, similar to the diagram presented in **Figure 25**.

Figure 27 illustrates this solution.

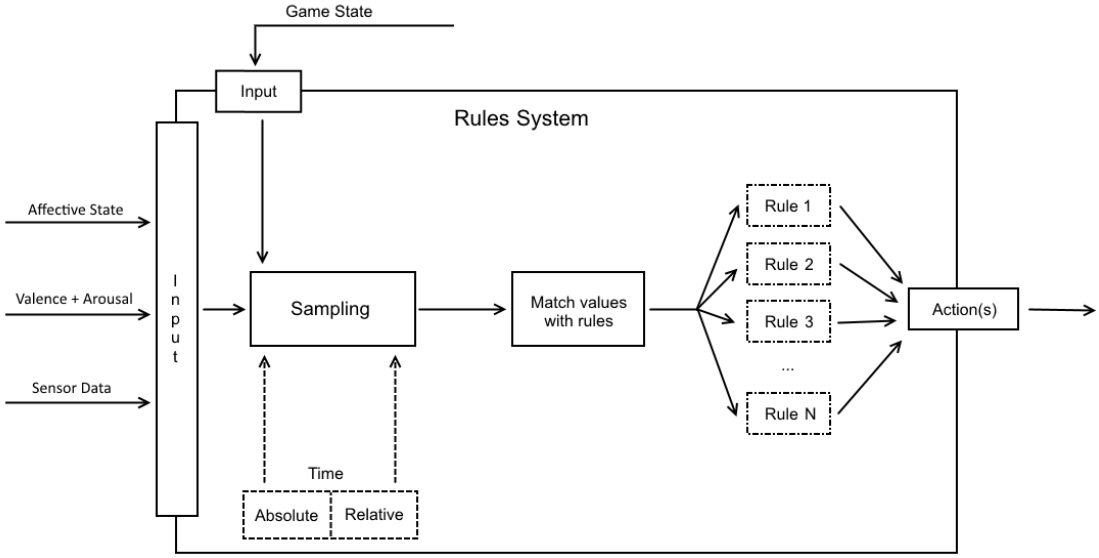


Figure 27: Rules system, solution 1 diagram.

With this solution, immediate comparisons, such as comparing one variable to a specific value set by the game designer, are possible as well is mapping biometric input to game mechanics. However, the game designer is not allowed to create rules that compare a variable to its previous value or state, somewhat restricting the type of rules that can be made. This would also restrict the implementation of profiling,[135] although a very limited implementation would still be possible, since this functionality might require a larger number of information (samples) in order to properly work.

3.2.2.2 Solution 2: Storing only one sample

In this second solution, we keep one sample stored (the previous captured sample) and continue to use the up-to-date, current captured sample to offer the same functionalities as the previous solution. This way, we are able to make comparisons between both samples, allowing

[135] For more information on profiling, see section **2.5 Profiling**.

to determine changes in values or states by iterations. Once this process is over (all rules are checked), the current sample overwrites the previous one and can be used in the next iteration.

Figure 28 illustrates this.

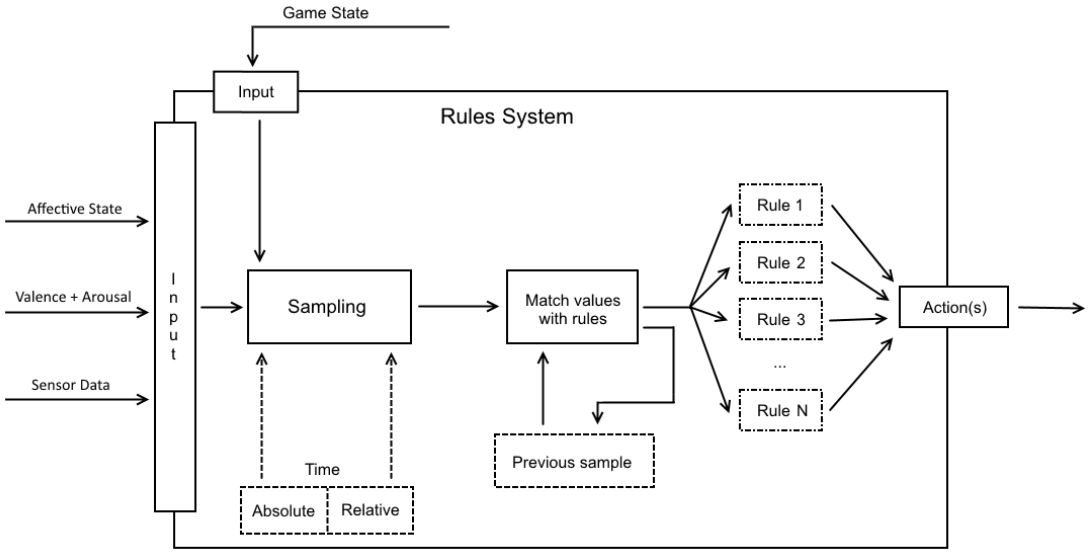


Figure 28: Rules system, solution 2 diagram.

This solution provides the same functionalities as the previous while expanding it with new possibilities. Rules that depended on the previous state of a variable can now be created. For example, it is now possible to create rule that states: "If the *current affective state* is 'happiness' and the *previous affective state* was 'sadness' then (do something)."

However, this solution does not yet provide the functionalities necessary to implement a detailed and powerful profiling tool since the number of samples available is still reduced. Another disadvantage is that, regarding the computational resources occupied by a sample, this solution will have higher requirements than the previous.

3.2.2.3 Solution 3: Storing a certain amount of samples

In this solution, the rules system stores the necessary number of samples – with this number being determined by the game designer[136] – in order to be able to perform its function. Once again, this solution is able to offer the same functionalities as the previous two and allows the implementation of a profiling tool, since the necessary number of samples are now available.

Figure 29 illustrates this solution.

[136] This number will depend on the complexity of the implemented rules and influenced by the machine performance (for example, the memory available will determine the maximum number of samples that can be stored). This way, providing a specific number for every scenario is impossible, and that is why the designer/developer has the ability to choose this value in their framework.

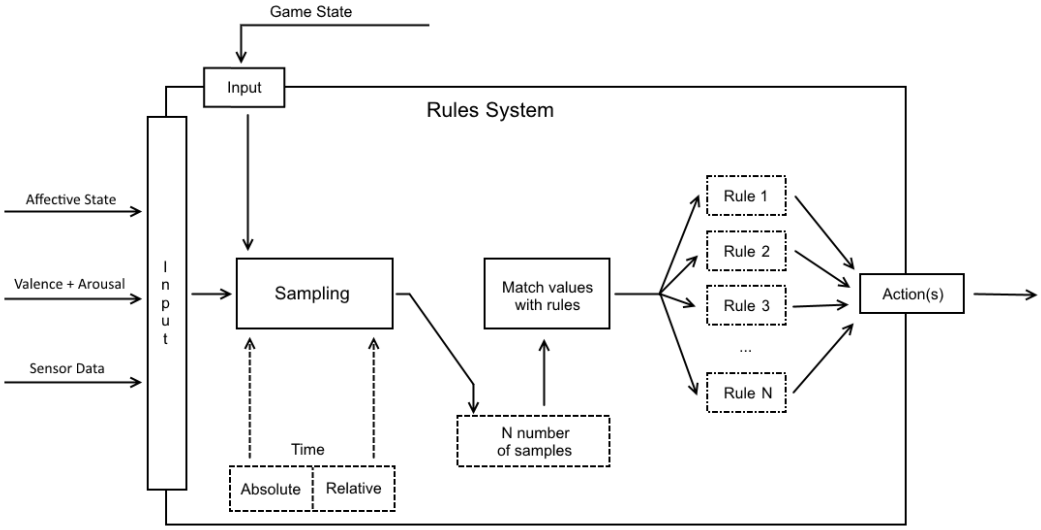


Figure 29: Rules system, solution 3 diagram.

However, this variant will surely demand more storage (or memory) space and processing power (due to the complexity and number of calculations that must be performed). For example, a rule that forces the system to calculate the average of the player's arousal may see the system's performance compromised: computing the average of 50 samples is drastically different than the same operation with 50.000 samples. This may impact the system's performance in such way that this solution becomes unfit for a real-time scenario. The game designer should take caution when choosing the number of samples.

3.3 Summary and conclusions

In this chapter we presented the conceptualization of the implemented framework, detailing its evolution from the core aspects to its interfaces, hierarchy and the relation between the components. The framework is composed by three elements: the component responsible for collecting and processing all the data the system will use; the rules system which receives these inputs and transforms them into actions, inputs understandable by the game, and the game itself.

This chapter also details the game mechanics used in *biofeedback* games, resulting from the research we did about the medium. To it, we add some suggestions though by ourselves with some examples backing them.

With the framework and rules system's structure detailed, it now must be given shape as a piece of software able to offer the desired functionalities. The next section will detail our implementation of the framework, along with the technologies used.

Chapter 4

Implementation of the framework and other support tools

This chapter will detail our implementation of the proposed framework and will be divided in three sections:

- The first will focus on the implementation of the biosignal collector and processor component,[137] its different layers and User Interface (U.I.) developed to configure them. We will call this component *Biometrics Input*.
- In the second, the functionalities developed in the game to communicate and receive information from the previous component will be detailed. We will also detail the type of structure developed to store the information and make it available to a game element that wishes to use these values. We will call this component *Biometrics Core*.
- The final subsection will focus on the implementation of the rules system, namely the in-game component created and the simplified notation language used to define the rules.

4.1 Biometrics Input

This component is responsible for collecting the biosignals captured by the sensors, transforming them into *arousal* and *valence*, and these into an *affective state*. This requires several steps making this component extremely complex in terms of implementation. We will try to address its implementation by level[138] in order to simplify the process.

[137] A detailed description of each component can be found in section **3.2**.
[138] The framework concept had three levels: the sensors, the arousal and valence and the affective state.

4.1.1 Technologies used

For the development of this component several technologies were used. **Table 7** present these, followed by its usage context and other details.

Table 7: Technologies used for developing the biosignals collector and processor component.

Technology	Usage	Other comments
Java Runtime Environment (JRE)[140]	This technology allows the execution of software created using Java as its programming language.	The choice of this programming language was done with the objective of creating a multi-platform solution,[139] supported by the many operative systems (OSes) as possible.
Java Development Kit (JDK)[140]	This technology allows the development of software created using Java as its programming language.	
Eclipse Integrated Development Environment (IDE)[141]	To further aid the development of the project, as it offered a set of important and useful tools.[142]	The author experience with this tool also contributed to its adoption.
Processing 3[143]	To ease the process of creating User Interfaces (UI) with an high degree of interaction and animation.	The Processing library as directly integrated in the Eclipse IDE in order to use the advantages and functionalities of both in the same environment.[144]
WindowBuilder plug-in for Eclipse[145]	This plug-in was used to create User Interfaces for forms and	It is used as a counterpart for Processing.

[139] Nowadays, Java is able to run in Windows, Mac OS and Linux operative systems.
[140] Both of the Java Runtime Environment and Java Development Kit can be downloaded at: http://www.oracle.com/te chnetwork/java/javase/dowloads/index.html (accessed June 20, 2016).
[141] Eclipse is a free and open-source software that can be download at: https://www.eclipse.org/downloads/ (accessed June 20, 2016). Eclipse Mars as the latest version at the time of the writing of this document.
[142] A source code editor, a debugger and a compiler are examples of these.
[143] Processing is an open source programming, with it origins in the Java language but with a simplified syntax. It is also an IDE. It can be download at: https://processing.org/download/ (accessed June 20, 2016).
[144] A tutorial on importing Processing into Eclipse can be found at: http://www.creativeapplications.net/processing/cra ft-your-processing-sketches-in-eclipse/ (accessed June 20, 2016). This tutorial, however, is for Processing 2, so the reader is also advised to read through the changes that took place. These can be found here: https://github.com/proces sing/processing/wiki/Changes-in-3.0 (accessed June 20, 2016).
[145] Instructions for installing, documentation and support can be found at: https://eclipse.org/windowbuilder/ (accessed June 20, 2016).

	menus that did not demand a higher level of animation.	
JavaScript Object Notation (JSON)[146]	Syntax used to store important values and configurations used by the framework in text files.	With the use of a library, this information can easily be read and written.
Json-simple (version 1.1.1)[147]	Library used to easily read and write data formatted under the JSON syntax.	Used to read and write the values and configurations necessary for the framework.
Open Sound Control (OSC) Protocol[148]	This protocol was used as a means of communications between the different components of the framework. This protocol also allow us to have the different components in the framework	Initially, this protocol was only meant to be used in the communication between this component and the game. However, at a certain point, it was also used communicate with the sensors.

4.1.2 Configuring the component

In order to allow the configuration of the Biometrics Input component, a set of user interfaces were created to aid the game designer or developer. These allow to register a set of sensors from which the physiological data of the player will be received; create and specify a circumplex model of affect; configure the IP addresses and OSC ports used by the framework to receive requests (from the game) and send the processed information (to the game) and generate the rules system. The interface for all these functionalities can be observed in the screenshot of the Biometrics Input start menu in **Figure 30**.

[146] JSON is a syntax for storing and exchanging data and an alternative to XML. Source: http://www.w3schools.com/json/ (accessed June 21, 2016).
[147] The library can be download at: https://code.google.com/archive/p/json-simple/ (accessed June 21, 2016).
[148] Open Sound Control (OSC) protocol was implemented using the oscP5 library, written by Andreas Schlegel, developed for the Processing environment. Like Processing, this library was also imported to use directly in Eclipse. Instructions for using and installing this library can be found at: http://www.sojamo.de/libraries/oscP5/#installation (accessed June 20, 2016).

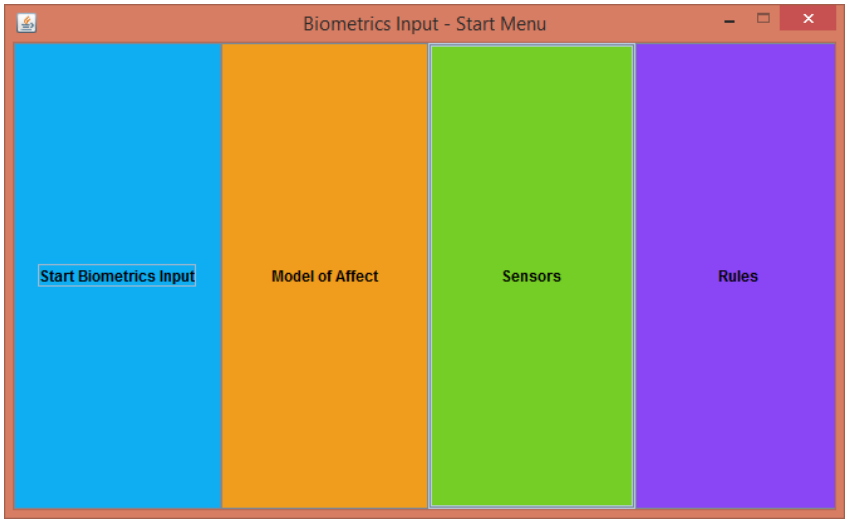


Figure 30: A screenshot of the Biometric Input start menu.

4.1.2.1 Configuring the component - Sensors

This component of the framework is responsible for registering sensors to be used in the game supported by the framework. By registering, we mean storing the pertaining information of a sensor (its name and type) as well as generate an OSC address that will be open to capture the data sent by it, if the sensors are indeed used. The game designer or developer can access the *Sensors* green separator (third separator from the left in **Figure 30**) to unlock the *Sensor Menu*, as seen in **Figure 31**.

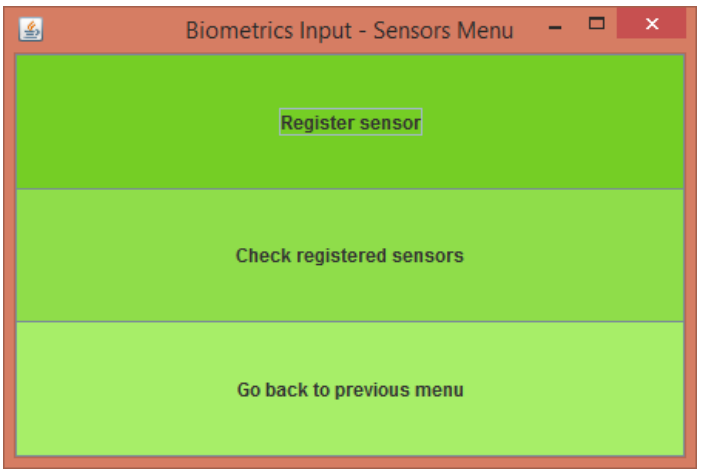


Figure 31: Biometrics Input, sensor menu screenshot.

To register a sensor, the game designer or developer needs only to fill the information requested, specifying its name[149] and type of sensor, as shown in **Figure 32**. The OSC address is automatically generated.

[149] The name of a sensor is not its branding or technical name but the way the game designer or developer desires to address the sensor inside the framework. For example, 'Microsoft Kinect 2.0' could be addressed as 'kinect' inside the framework.

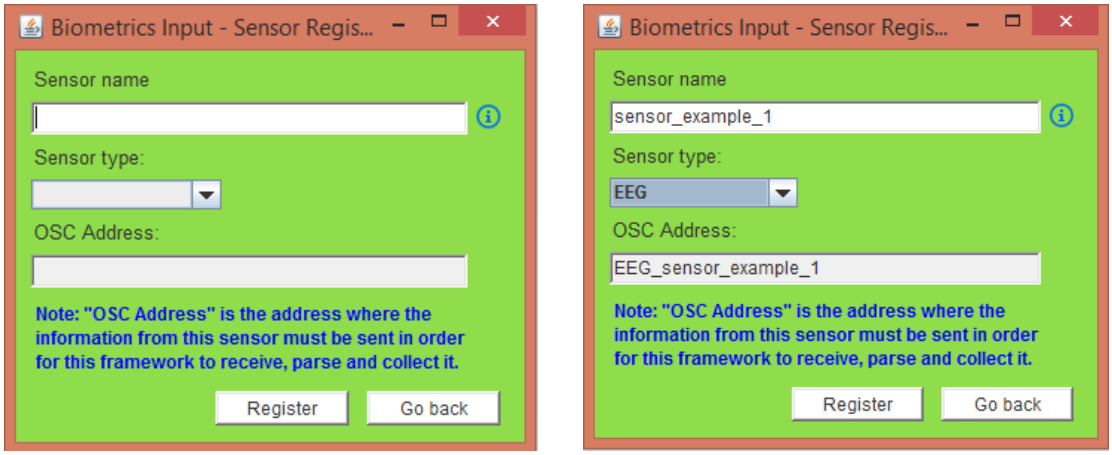


Figure 32: Registering a sensor and automatic generation of the OSC address.

The name of the sensor must be composed only by letters, numbers, spaces or underscores; other special characters will be automatically removed. Also, the type of sensors available are restricted to the values[150] presented in the dropdown list. Once this process is complete (and the 'Register' button pressed) the sensor data is stored in a text file, strucutured according to JSON notation.

It is also possible to consult the registered sensors in the framework by selecting 'Check registered sensors'. **Figure 33** presents this menu graphical interface. Only one sensor is presented at a time. If more than one is registered, the menu will offer the possibility to shift to another by using the arrows located in the bottom centre.

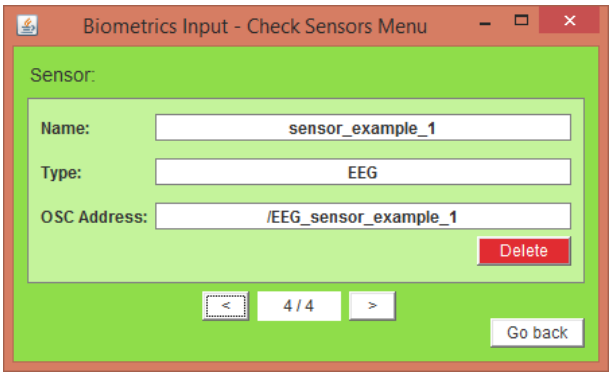


Figure 33: Screenshot of the menu that presents the registered sensors information.

The game developer or designer can also delete any undesired sensors by pressing the red 'Delete' button. As this is an irreversible action, a confirmation window will pop-up, asking the game designers to revalidate their intentions, as seen in **Figure 34**.

[150] These values are: 'EEG', 'BREATHING' , 'EYE_MOVEMENTS', 'EMG', 'FACIAL_EXPRESSIONS', 'ECG', 'SCL_SRL', 'TEMPERATURE', 'OXYGEN', 'POSTURE', 'BLOOD_PRESSURE' or 'OTHER'. This options mimic the biometric data presented in section **2.3.4**.

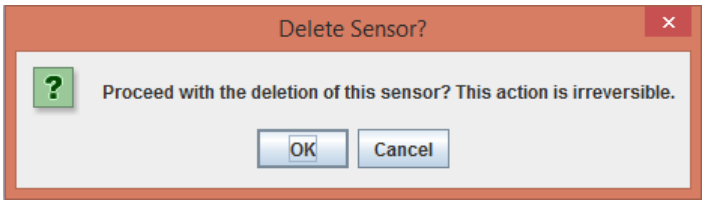


Figure 34: A confirmation window asking the game designers to validate their input in a irreversible situation.

Throughout the framework, whenever the user is faced by an irreversible action, a similar confirmation window will be presented. However, as they are very similar (only the message content changes), we show only one example.

Finally, one important point to highlight is that although the type of sensor is registered, this information is currently only being used for the generation of the OSC address. We felt that this information should be stored as it could possibly be used in future iterations of the framework.

4.1.2.2 Configuring the component - Arousal and Valence

In order to determine an affective state, the arousal and valence must first be extracted from the biosignals captured by the sensors. In order to achieve this, we intended to replicate a sub-system of Nogueira's framework,[151] namely the *Physiologically-Inductive Emotion Recognition Sub-system* (PIERS). Unfortunately, due to time restraints, it was not possible.

However, in order to permit the remaining components of the framework to be tested, we created a special module that would simulate the intended functionalities of this component. This module would generate a random value[152] for arousal and valence that would vary smoothly through time. This value varied between -1[153] and 1[154] and always start at 0. The generated values would then be mapped to a two dimensional model, with abscissa axis mapped to valence and the ordinate to arousal, with a combination of these two resulting in an affective state. This affective state dependended on the circumplex model of affect selected[155] by the game designer or developer and where represented as colours.

To better visualize the previous information and monitor the evolution of these values, we extended the previous component and added a graphical interface, which can be observed at **Figure 35**. In it, the selected circumplex model of affect is drawn in the centre, along with its limits (black line surrounding the model) and the arousal and valence axis. In the bottom, a bar contains information regarding the current and previous affective state[156] as well as the arousal

[151] This framework, and its components, have been described in section **2.4**.
[152] With this value varying between -1 and 1.
[153] Representing the most negative (for valence) or most passive (for arousal) affective state.
[154] Representing the most positive (for valence) or most active (for arousal) affective state.
[155] More information about the selection of the circumplex model of affect will be given in section **4.1.2.3**.
[156] Affective state is referred as emotion in the interface.

and valence value. A reddish cross marks the current arousal/valence coordinates[157] in the model.

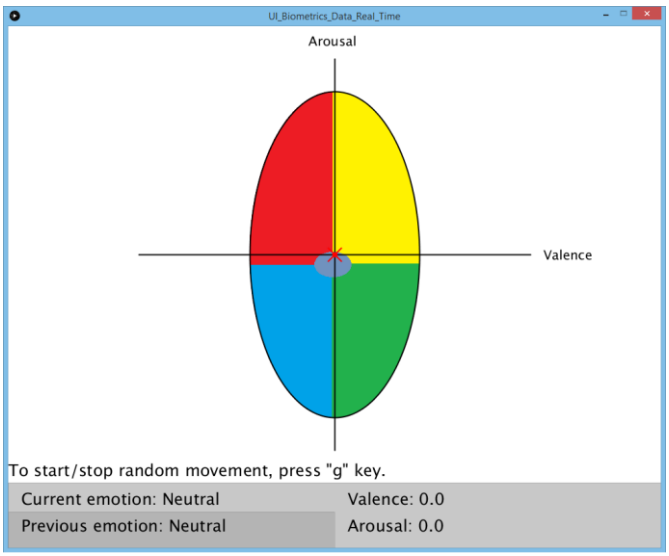


Figure 35: The valence and arousal generator module graphical interface.[158]

One important point to remember is that generation of these values is arbitrary, which makes reaching a specific affective state or arousal/valence value a very difficult challenge. In order to avoid this undesired situation, we mapped the mouse position to the coordinates of the referential and, when the game developer or designer presses the mouse, those coordinates immediately become the current arousal and valence value. Consequently, the affective state is also changed to match the new values. If, however, the mouse selection was outside the model, the closest point to that position, inside the model is selected. We also added the possibility of stopping the generation by pressing the 'g' button, removing fluctuation in the values selected using the mouse.

4.1.2.3 Configuring the component - circumplex model of affect

As presented in the conceptualization of the framework, section **3.2**, one interface is responsible for converting the arousal and valence values, given by the layer bellow, into an affective state. In order for this to happen, the framework requires a circumplex model of affect to match these values with a state.[159] As there is not a generic affective model to apply to every situation, and forcing the game designers and developers to use one model developed by us seemed an incorrect and restrictive attitude, we decided to create a tool that allows the creation

[157] In other words, the point's X coordinate matches the valence value and the Y coordinate matches the arousal value.
[158] Although just an example, the model being used in this image is not a perfect circle reflecting the possibility of the circumplex model of affect being shaped differently when applied to children, as previously discussed on section **2.2.1**. The maximum and minimum axes values are also adapted in order to match the model: with the Y axis still ranging between 1 and -1 and with the X axis varying between 0.5 and -0.5.
[159] Or, put another way, the framework requires a set of rules which it can use to map the arousal and valence into a specific affective state.

of customizable circumplex model of affect. This tool was later fused with the Biometrics Input becoming one of its functionalities.

The circumplex model of affect menu can be accessed by selecting the respective separator in the start menu (**Figure 31**). This new menu, as seen in **Figure 36**, will offer the game designer and developer the possibility of creating a circumplex model of affect, upload it to the framework and consult other updated models.

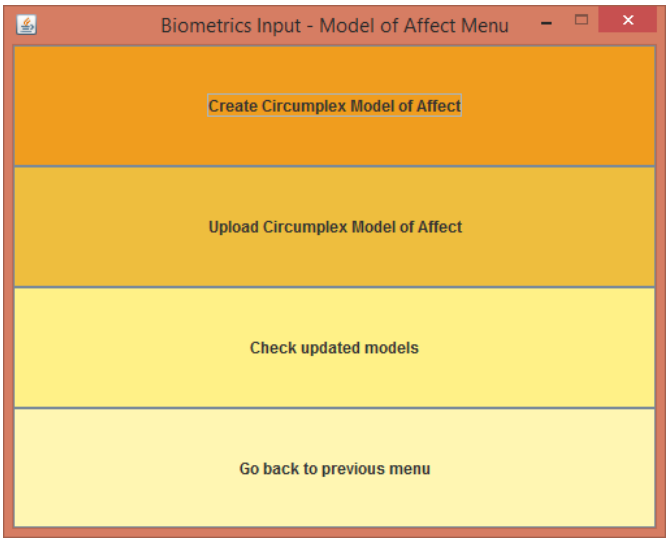


Figure 36: Screenshot of the Biometrics Input (circumplex) model of affect menu.

The process to create a new model is the following: 1) decide the dimension of the circumplex model;[160] 2) paint it using an external tool, with each colour representing an affective state; and 3) upload to the framework, matching each colour with its intended meaning.

To achieve the first step, the game designer and developer need to select 'Create Circumplex Model of Affect' (whose screenshot can be seen at **Figure 37**) and define the arousal and valence dimensions. To input these, the game designer and developer need only to press the white squares with coloured numbers located in the bottom part of the interface, with the blue numbered box allowing to arousal value and the red numbered box the valence. When pressed, an input interface will pop-up, asking for a new value for the chosen dimension. This value must vary between 100 and 999,[161] as seen in **Figure 38**. Changing a dimension is

[160] How many pixels will the valence and arousal dimension possess.
[161] These values were randomly chosen. More tests are required for determining a suitable minimum and maximum value or the removal of these limits.
One important point to highlight is that these dimensions will affect the model's maximum and minimum axes values. If the model has the same dimension for the X and Y axis then the maximum and minimum values will range between 1 and -1 for both. If, however, one axis has twice the dimension of the other, then that axis value will range between 1 and -1 while the other will range between 0.5 and -0.5. In other words, the biggest axis (the axis with the highest dimension) will always range between 1 and -1 while the smallest will suffer the necessary transformation.

equivalent to changing the circle[162] radius. The higher the dimension the higher the resolution of the model.

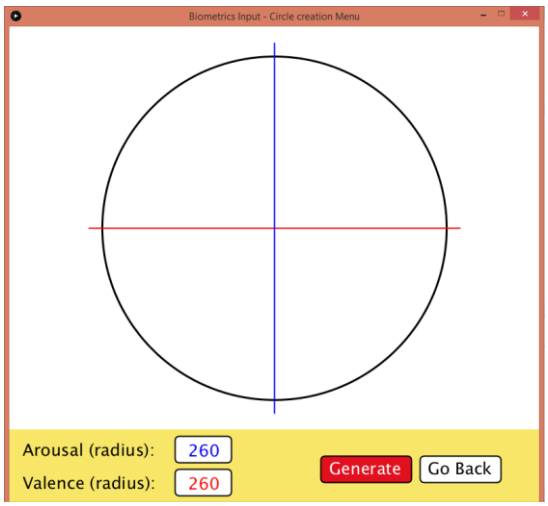


Figure 37: Generating a circumplex model of affect with the desired dimensions.

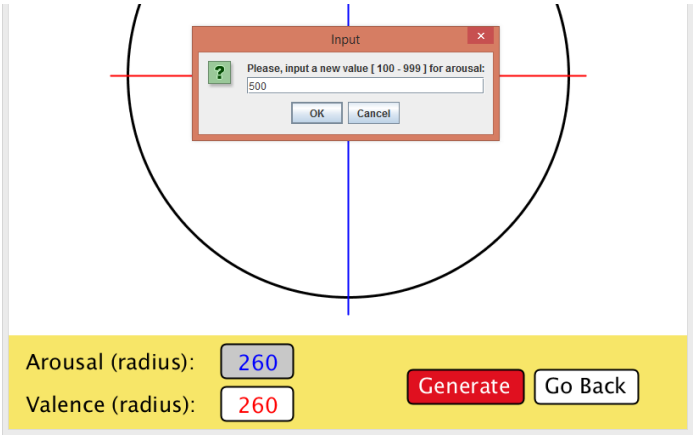


Figure 38: Specifying a new value for the arousal dimension,

When satisfied with the circle size, the game developer or designer should press the 'Generate' button. The system will then request a directory for storing the results: a *.png* file with its width and height matching the arousal and valence values, respectively.[163] The game developer and designer would then need to colour this circle using an image edit tool. An example of this process can be seen in **Figure 39**.

[162] Correctly speaking, this circle should be called an ellipse, as a circle has the same radius for the X and Y axis, while an ellipse does not.
[163] In other words, if the arousal dimension is 250 and the valence dimension is 300, the resulting image file would be 250 (width) per 300 (height) pixels in size.

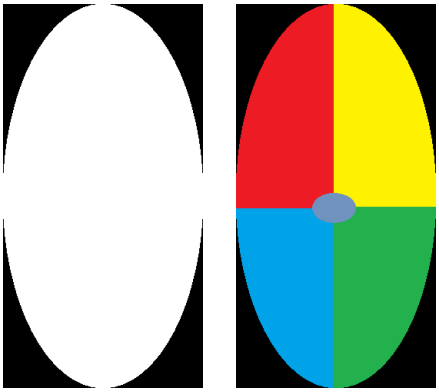


Figure 39: The resulting image file before (left) and after (right) the colouring process.

The black coloured area observable represents the area outside the circle, which should not be filled as all the information in the it will be ignored. As a restriction, the game designer or developer are also not allowed to use the black colour as it will also be ignored in the uploading process.

Once satisfied with the result, the image can be uploaded back to the framework. To achieve this, the game developer or designer must select the 'Upload Circumplex Model of Affect' (**Figure 36**) and use the adequate button[164] to select the image file. This menu interface can be seen in **Figure 39**.

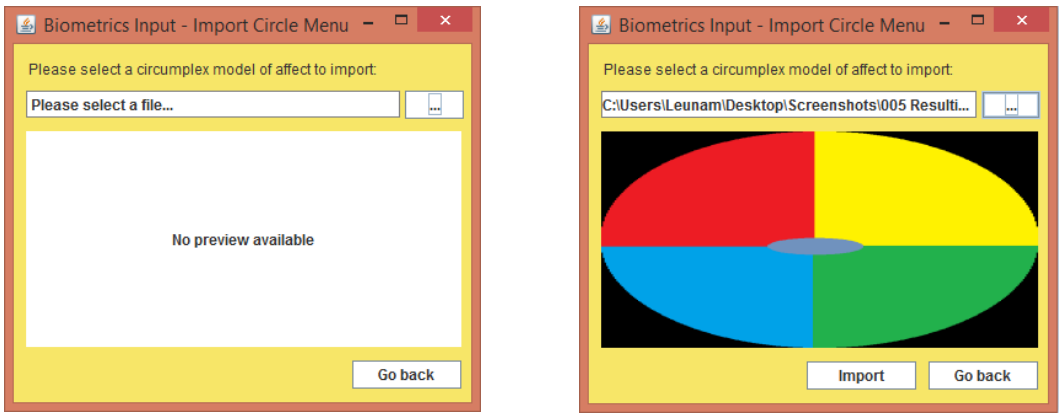


Figure 40: Import circumplex model menu before (left) and after (right) the import.[165]

Once the import is complete, the game designer or developer still need to assign a meaning to each colour. This can be done in the new menu that pops-up, shown in **Figure 41**, by filling the text fields presented and press the 'Apply' button. This will assign the written value to the colour present in the square while also checking if the value assigned is valid[166] and unique.[167] If

[164] The rectangular button with the '...' symbol.
[165] The imported image is compacted in order to fit inside the preview window. Its dimensions, however, remain the same.
[166] For a value to be valid it must be a string of number or letters, without any special characters nor spaces.
[167] The same value is not repeated in another colour. In other words, there are not two affective states with the same meaning or name.

unsatisfied with the meaning given, the 'Edit'[168] button can be pressed to edit the text field contents.

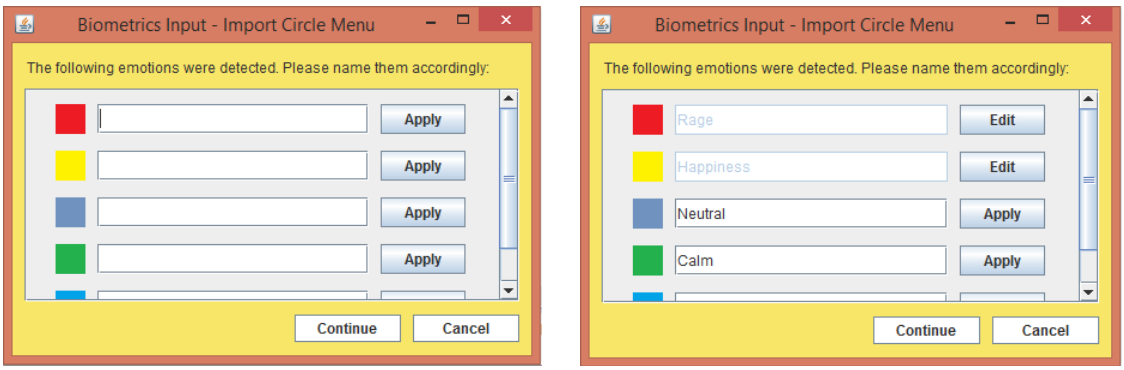


Figure 41: The process of giving meaning to each colour.

Once all the text fields are filled and validated, the 'Continue' button can be pressed to conclude the upload process. This information is stored in the framework in two ways: 1) all the affective states colours, meanings, circle dimensions and ID[169] are stored in a text file, structured accord to the JSON notation, and 2) the image file is stored in a folder, created exclusively for the purpose of storing the models' images, so that the system could load these whenever necessary.

The final functionality provided by this component is the possibility to review previous uploaded circumplex model of affects. The graphical interface is similar to the one used in the *Sensors* components, also offering the possibility of browsing through the stored models and delete them, if necessary. A screenshot of this interface can be seen in **Figure 42**.

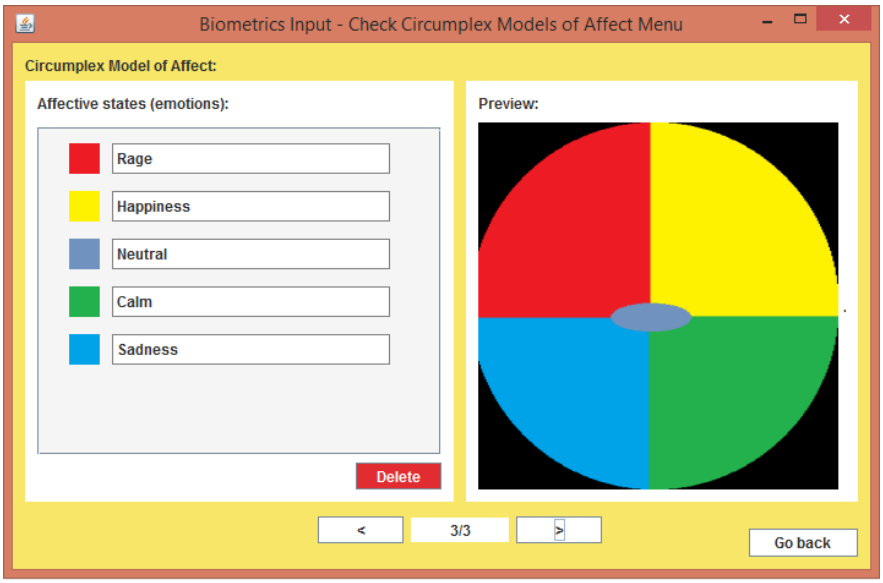


Figure 42: Screenshot of the 'Check Circumplex Models of Affect' menu.

[168] The 'Edit' button takes the place previously occupied by the 'Apply' button, as can be seen in **Figure 41**.
[169] This ID is assigned by the system to the model during the upload.

4.1.2.4 Configuring the component - globally

Now that all the Biometrics Input components have been presented,[170] all that remains is tying them together. To ease this process a graphical interface was created, as can be seen in **Figure 43**.

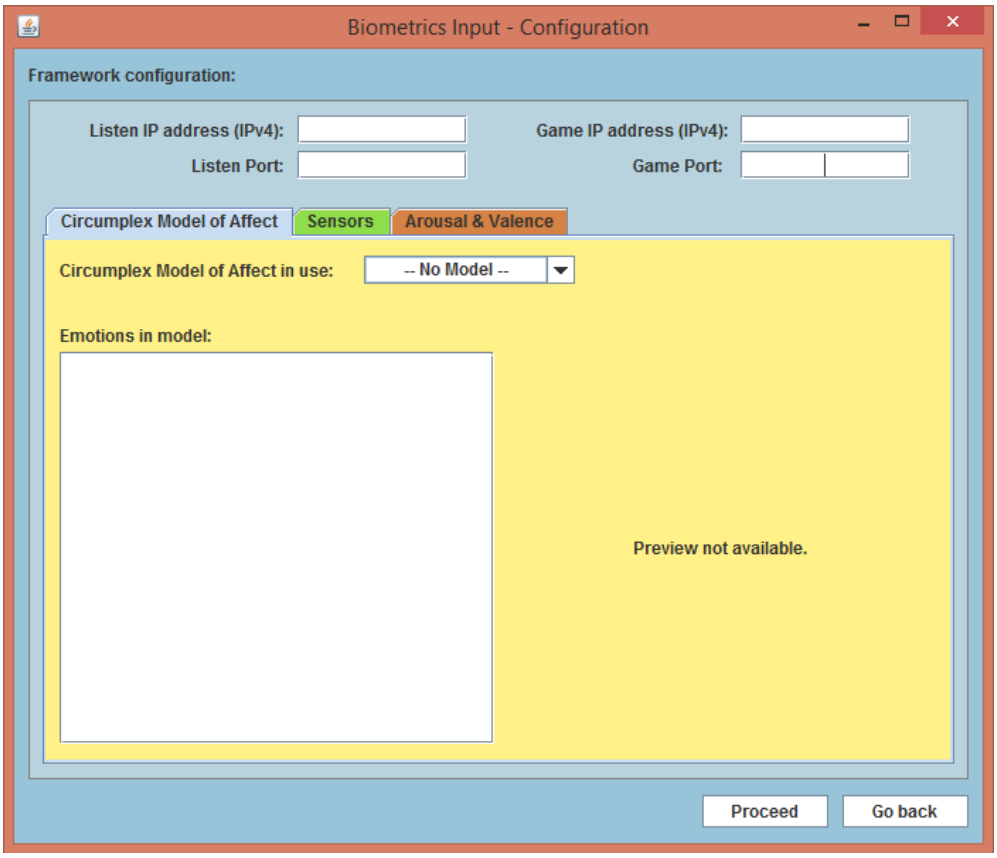


Figure 43: Screenshot of the Biometrics Input global configuration menu.

In here the game developer or designer must provide the following information: 1) which port[171] will the framework listen for requests,[172] 2) which IP address and port should the information collected by the Biometrics Input be sent to[173] and 3) which model and sensors are to be used in the process. A tab has been created for every customizable component[174] and the selection process is illustrated in **Figure 44** and **Figure 45**.

[170] Although included in the Biometrics Input, the rule system generator is considered an external element. However, for the sake of centralizing all the tools developed, it was appended to this component.
[171] There is no need to specify the 'Listen IP address' as the framework will use the IP address of the machine running it.
[172] These requests are sent by game during the start up process. A request example the game asking the Biometrics Input how many affective states are being used and which are they.
[173] In other words, the machine's IP address where the game is running from and the port that it is listening to.
[174] Although the 'Arousal & Valence' tab is empty.

To select the circumplex model of affect, the game designer or developer can make use of the drop-down list enclosed in the tab 'Circumplex Model of Affect', which contains all the models available in the framework.

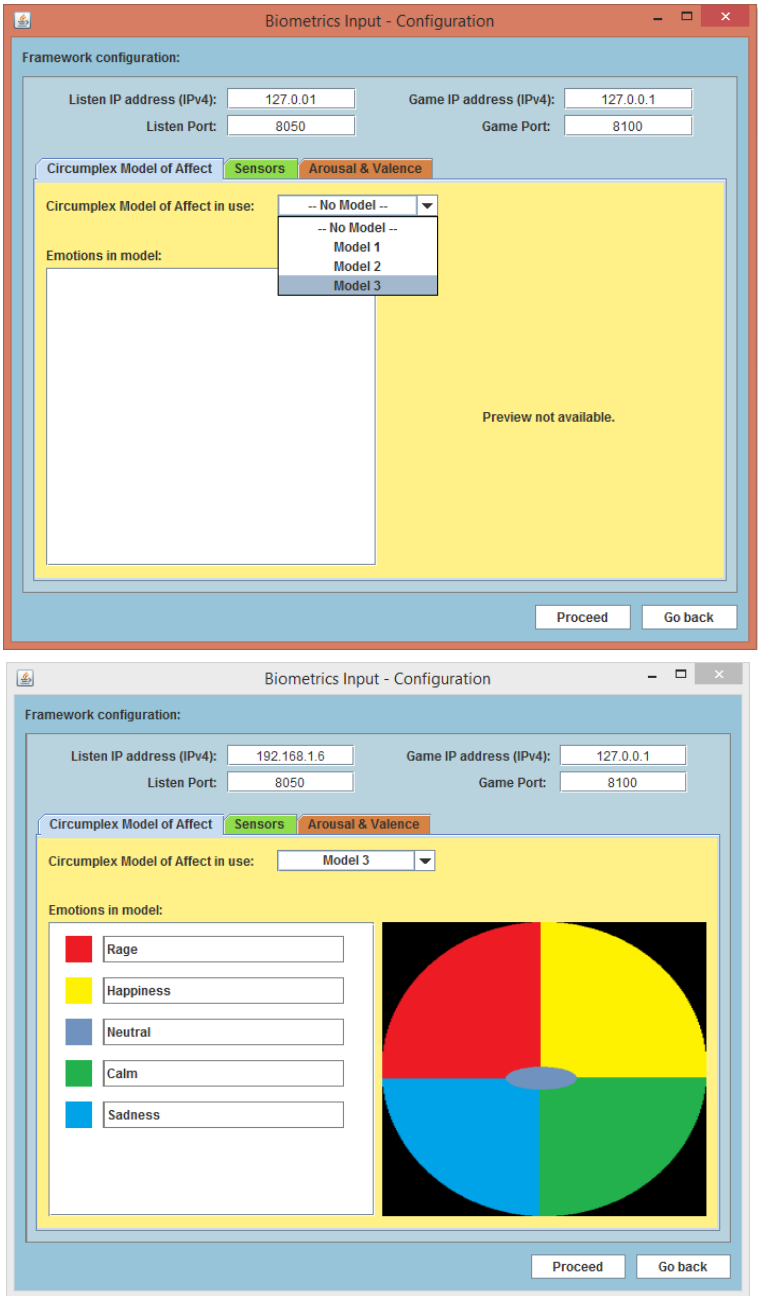


Figure 44: Selecting a circumplex model of affect.

For selecting the sensors to use, the game designer or developer can make use of the functionalities provided by the respective tab. In it there are two lists, with the left list showing which sensors are to be used and the right containing all the registered sensors. In order to swap

one sensors from one list to the other, the game designer or developer need only to select the sensor and press the swap button,[175] making the sensor swap lists.[176]

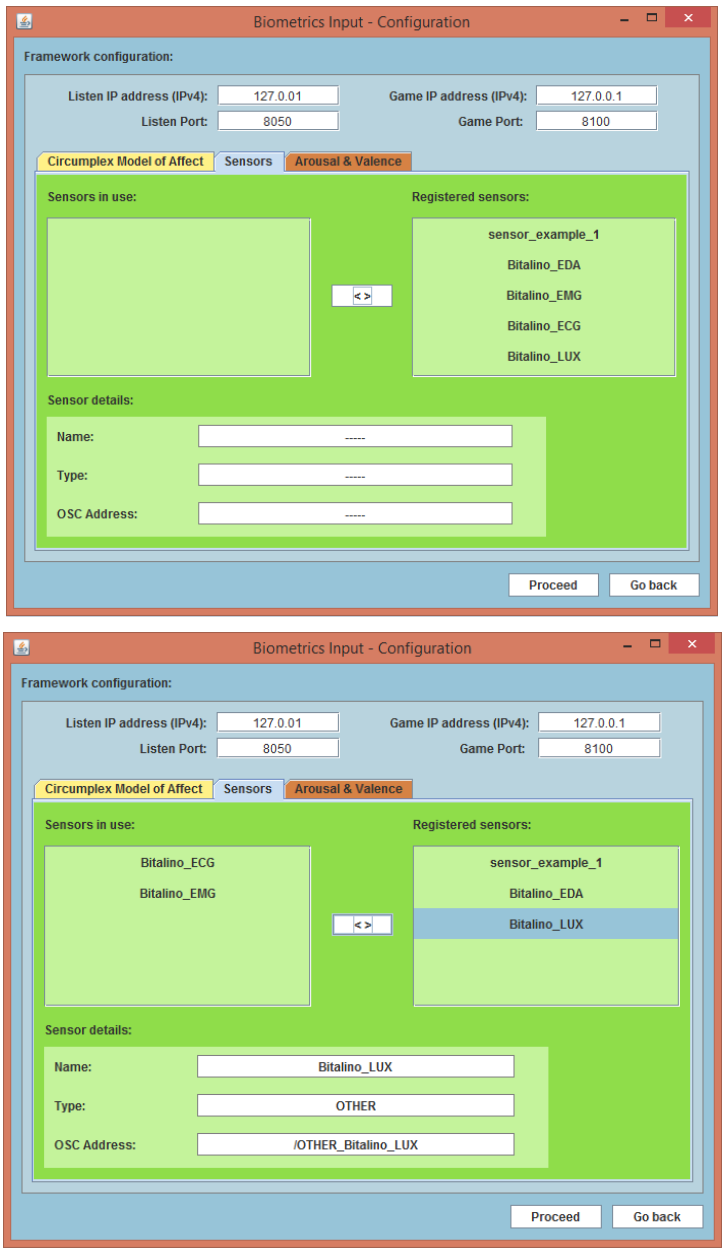


Figure 45: Selecting which sensors to use.

Once satisfied with the configurations, the 'Proceed' button should be pressed to end the configuration and start up the Biometrics Input component. This would bring the game developer or designer to the arousal and valence generator component.

[175] Button containing the symbol '< >' (see **Figure 45**).
[176] If the sensor has contained in the list 'Registered sensors' list, pressing the swap button would result in it being swapped to the 'Sensors in use' list, with the reverse also being true.

One point to highlight in this menu, is that during tests there was a tendency for frequently leaving and entering it,[177] having to insert the configuration (IP addresses, ports, models, sensors) each time this happened. In order to avoid this distressing situation, we decided to save the inputs into a file anytime this menu was abandoned, loading them whenever it was reopened. This file was also structured using the JSON notation as we could reuse the technology employed for reading the sensors and models information. While initially temporary, this ended up becoming a configuration file, which would be read by the Biometrics Input when starting up.

4.1.3 Flow and inner workings

Based in the configuration process described in the previous section, the framework flow until this point can be summarized in the diagram presented in **Figure 46**.

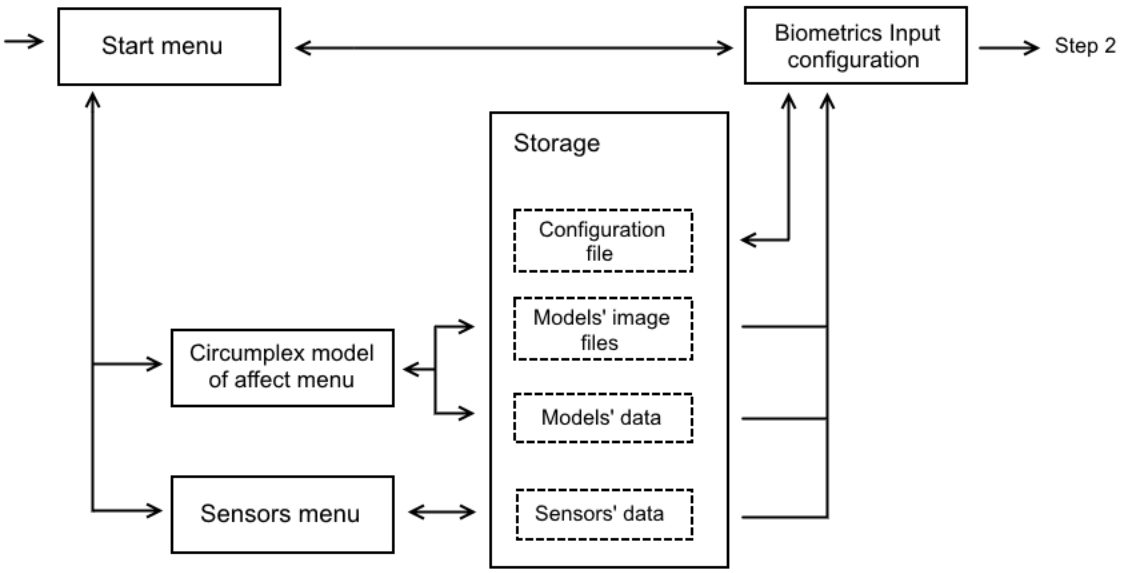


Figure 46: Initial framework flow.

With the Biometrics Input configuration complete, it can now be ran using the chosen settings (Step 2). Before running, however, it was to: 1) open an OSC communication channel;[178] 2) start the component responsible for dealing with the sensors data; 3) start the arousal and valence generator, 4) load the circumplex model image and mechanism that will match the values to an affective state. Only then, it can start communicating with the framework in the game side.

Each of the previous steps as an objective. The OSC communication channel is used for three actions: communicate with the sensors (to receive their values), transmit all the

[177] This happened because it was frequently needed to add a new sensor or a new model.
[178] Using the IP address ('Listen IP address') and port ('Listen port') specified in the configuration file

information gathered (sensor data, valence, arousal and affective states) to the game[179] and to receive and reply to requests made by the game.[180]

The sensor component, on the other hand, uses the OSC communication channel previously established to capture the physiological data sent by the sensors. These data is then stored and made available to all the other components that may require it. This information is updated any time the respective sensor sends a new value, which could vary.

The arousal and valence generator will generate a dummy value for both these values. As the sensor data, these are also stored and accessible to any component that may require them. Their update rate of depends on the Processing number of frames per second,[181] as these are generated in the *draw()* function.

The mechanism in charge of matching the arousal and valence with the circumplex model of affect uses the values stored by the previous component to determine the current affective state. To achieve this, the mechanism works as follows: given X arousal and Y valence, these coordinates are looked up in the model and the pixel colour at the said position is read. The affective state is straightforwardly determined by matching it with the colour given with meaning from the model. The result is stored as well and the updating rate of this component is also dependant on Processing's frame rate.

Finally, the Biometrics Input gathers all these values and sends them to the game. It is also responsible for replying to any requests that arrive through the OSC communication channel.

With this information, the initial flow diagram could be updated, which the reader may find in **Framework flow (full diagram)**.[182]

4.2 Framework (game side)

In the previous section, we focused on describing the Biometrics Input, an element of the framework responsible for collecting the data that will be used as an input by the game. However, this data cannot simply be sent and expected to work immediately. In order to receive it, unpack it and translate it into an input understandable by the several more components had to be developed and implemented in the game side. Their names are Biometrics Core, Biometrics Receiver and Biometrics Log, which will be addressed in detail in this section.

[179] The time between each transmission of data is a little above sixty times per second. However, this value can reduce depending on the resources (processing power) the machine has.
[180] These requests, sent during the setup of the framework on the game side, are made with the purpose of discovering how many sensors are being used, what are their names and what affective states available in the model being used.
[181] Default value is 60 frames per second. Source: https://processing.org/reference/frameRate_.html (accessed June 29, 2016). Depending on the machine's resources (processor power) in which Biometrics Input is running, this value could be lower.
[182] This was done because the diagram exceed the page size and its resized version was illegible.

4.2.1 Technologies used

For the development of these components, different technologies from the Biometrics Input were required. **Table 8** details these, accompanied by the reason they were chosen (their usage) and other details.

Table 8: Technologies used for developing the framework components used inside the game.

Technology	Usage	Other comments
Unity3D[183]	Unity is a cross-platform game engine. Just as Java as used to develop a multi-platform solution, it would only be fair that the game that used it could also run in several system.	This specific game engine was chosen due to our previous experience with it, something we hoped would speed the framework implementation.
Unity OSC[184]	This protocol was used as a means of communications between the Biometrics Input and the components of the framework inside the game.	-----
C# (C sharp)	C# was the programming language chosen for the creation of the framework scripts.	Unity also allowed Javascript for this task.

4.2.2 Biometrics Core

Biometrics Core intended to be the central and most important component of the framework inside the game. This component is responsible for opening an OSC communication channel with the Biometrics Input and exchange information with it. By information exchange we mean request the necessary information to correctly initialize the framework[185] and receive the packages of information sent during the game.

A screenshot of this script can be seen **Figure 47**. In order to properly work, Biometrics Core must informed which port to listen to[186] and which IP address and port to talk to.[187] This

[183] For more information on the game engine, the reader is redirected to its official page at: https://unity3d.com (accessed June 29, 2016).
[184] Open Sound Control interface for the Unity3d game engine, Copyright (c) 2012 Jorge Garcia Martin.
[185] In Biometrics Input, this information exchange was described as a request from the game.
[186] In Biometrics Input, this port was called 'Game port' in the configuration menu.

script also offers the functionality of saving a log with all the information received during play. It is disable by default but can enabled by simply checking 'Use Biometric Data Logger' box.

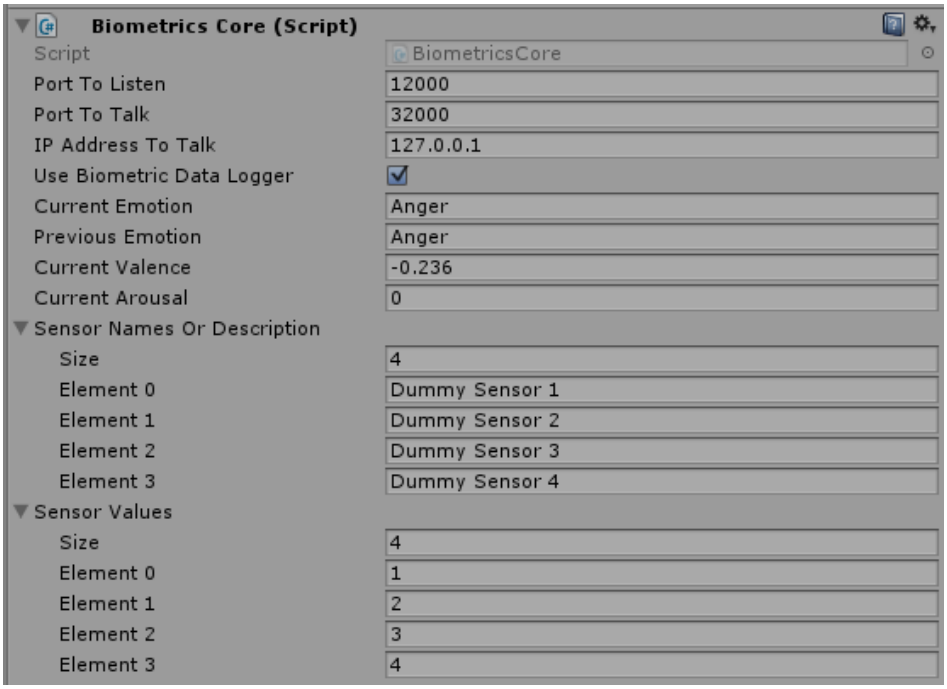


Figure 47: Screenshot of the Biometric Core script.

All the Biometrics Core functionalities are performed by an auxiliary class. This class creates the communication channel, listens for messages from Biometrics Input and unpacks the information sent. All the information received is displayed in the script in real-time. For example, the variable 'Current Emotion' contains the last affective state received. 'Sensor Names Or Description' is a list with all the names or descriptions of the sensors in use.

One important point to highlight is that, in order for the framework to properly work, this script should be contained inside a game object tagged 'BiometricsCore' and only one game object with this tag should exist.

4.2.3 Biometrics Receiver

Biometrics Receiver is a script that contains the same information as the Biometrics Core and is continuously updated by it. This script contains an abstract class, with the same name, that the game developer may extended to have access to the information sent by the framework. This way, Biometrics Core remains unaltered, and the game developer can freely experiment with data without risking breaking the framework.

[187] These are the IP address and port of Biometrics Input: 'IP Address To Talk' and 'Port To Talk' in the configuration menu.

This solution is, nevertheless, still very technical and restricted to those with programming knowledge. To counter this, we developed the rules system, which was already detailed in section **3.2.1**, with its implementation to be explained in section **4.3**.

4.2.4 Biometrics Logger

Biometrics Logger is the script invoked by the Biometrics Core if the game designer or developer desire to make a log of all the information sent by Biometrics Input, either for debugging or studying purposes. All the data is stored in a text file, with the file path presented in the script, as can be seen in **Figure 48**.

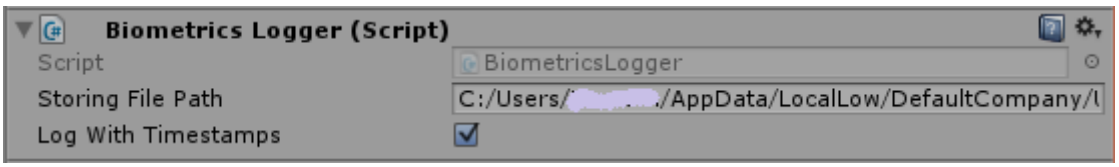


Figure 48: Biometrics Logger script.

The script also offers the possibility of logging with timestamps. This adds the time each sample[188] was collected, in Biometrics Input, and the time the same sample was received, by the Biometrics Core, to the text file. An excerpt of a log file is given in **Biometrics Logger excerpt**

4.2.5 Installation process

Once the development of Biometrics Core was completed, we exported it as an *unity package file*: a compressed file containing all the assets created. This file can be executed by double-clicking on it, with the engine importing its contents immediately into the current open game project. In order to speed even more this installation, we create a *prefab* of the Biometrics Core *game object*, which we include in the package, so that the developer only needs to drag and drop it in the hierarchy. Importing this package results in a folder, name Biometrics, and the said prefab, also with the same name, as can be seen in **Figure 49**.

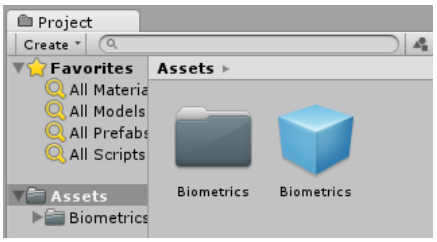


Figure 49: Resulting files from unity packege import.

[188] A sample is the information sent by Biometrics Input at an specific moment.

The folder contains several important scripts (among these are Biometrics Core, Biometrics Logger and Biometrics Receiver) critical to the framework and, as such, we recommend that they are not moved, edited or deleted.

4.3 Rules system

With the framework implemented and ready to be used, the flow of the biometric data is now possible. However, the game designer still has to determine how these values are going to be used and, once complete, the game developer still has to implement the solution. This would mean extending the Biometrics Receiver, and use the values provided by it to implement the game mechanics matching the game designer view.

In the hope of speeding this process, we devised the *rules system*, which is: a simplified notation language, simple to use and easy to learn (hopefully), and a code generation tool. The notation language provided a structure and syntax that the game designer can use to create the game mechanics that required biometric data as their input. These *rules* are comparisons between values that, when true, will trigger an action in the game, influencing it in some way. For example, a rule could be *when the average arousal is lower than 0.5 then action5 would be performed.* The code generation tool parses the document, created by the game designer using the said language, and, if no errors are found, automatically generates a C# script. This script contains the same rules (contains the same comparisons) and actions as the original document but written in a programming language compatible with the Unity3D game engine.

4.3.1 Technologies used

In an attempt to centralize all the framework tools into a single space, the rules system ended up being included as functionality of the Biometrics Input. As such, the rule parser[189] and code generator[190] (both belonging to the rules system) ended up using the same technologies used in Biometrics Inputs, so that this integration is possible. The simplified notation language is in fact a XML document structured according to specific rules defined by us.

With this, **Table 9** details the technologies used for developing the rules system.

Table 9: Technologies used for developing the rules system.

Technology	Usage	Other comments
Java	We used the same programming language as the Biometrics Input so that the rules	---

[189] The program that reads the rules defined by the game designer and evaluates if the structure and syntax are correct.
[190] The program that transforms the rules into code usable by the game engine.

(Programming language)	system could be appended to it seamlessly.	
eXtensible Markup Language[191] (XML)	Served as the basis, and is the core, of the structure of the simplified notation language we developed.	The structure and other technical details will be explained in section **4.3.4.**

4.3.2 The process

In order to properly use the rules system the first step is writing and XML document, matching the structure defined by us, specifying the desired rules in it. Once completed, this document must be feed to the rules system parser and code generator in order for the creation of the C# script to happen. To do this, the game designer or developer can access the rules separator, in the start menu of Biometrics Input (**Figure 30**), to open the graphical interface that will simplify this process. A screenshot of this menu graphical interface can be found at **Figure 50**.

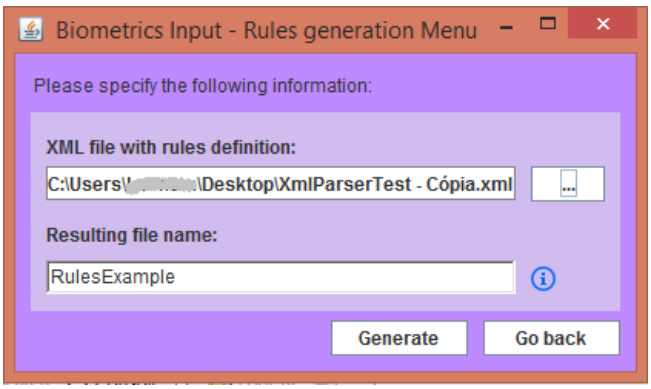


Figure 50: Rules system (rules generation) menu.

To use this interface, simply select the XML file containing the rules,[192] name the C# script (in **Figure 50**, it was named *RulesExample*) and press the 'Generate' button. If no errors are found by the parser, the system will output the resulting C# script at the same directory as the input file.

One important point to highlight is the naming process. The name specified will be given to the C# script and to the class contained inside it. For example, if we were to name our script *RulesExample*, the system would output a file named *RuleExample*.cs, containing the class *RulesExample*.

[191] XML is a markup language, readable by both humans and machines, designed to store and transport data. Source: http://www.w3schools.com/xml/ (accessed June 30, 2016).
[192] This is achieved by pressing the '...' button. This action will open a file browser so that the game designer or developer can select the desired file.

4.3.3 C# script

One of the tools offered by the rules system is the code generator. This tool outputs a C# script, containing the rules defined by the game designer, and a set of empty functions, at the end, matching the actions defined in the XML document. For example, the rule *if the current valence value is higher than 0.5 then do changeAvatarSpeed* would generate the code contained in **Figure 51**.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Biometrics;

class RulesExample : BiometricsRulesetBase
{
    protected override void checkConditionsAndTriggerActions()
    {
        if( getCurrentValence() > 0.5f)
        {
            changeAvatarSpeed();
        }
    }

    private void changeAvatarSpeed()
    {
        //Write the desired code for this action.
        throw new NotImplementedException();
    }
}
```

Figure 51: Class outputted by the rules system code generator.

As can be seen, the rule was converted to an *if condition* and the action to an empty function[193] with the same name. To implement this script, the game developer or designer need only to import it into the game engine and apply it to a game object. The script will automatically configure itself, connect with Biometrics Core and perform its functions. What it will not do is create the code inside the generated functions: this must be done by the game developer and match the game designers intentions.

Another point worth highlighting is the extension of the abstract base class *BiometricsRulesetBase*. This class was created with the intent of containing the common operations of every generated script, leaving the extended classes less cluttered. This way, the base class will automatically connect itself to the Biometrics Core, do its own set-up, collect samples at a set rate,[194] evaluate the rules and perform the respective actions.

By collecting samples, we mean that the base class will access Biometrics Core, every frame, copying all the biometric data at the specific moment. This information is then stored inside the class until the maximum number of samples is reached. When this happens the oldest

[193] Technically, it is not empty, as it throws an exception. We decided to throw a *NotImplementedException()* to remind the game developer and designer that the action must be implemented by them.
[194] The collection of samples happens at the same rate as the Mono Behaviour function *Update()*. This means that a new sample is collected each frame.

sample will be discarded, creating the necessary space for a new one. The limit of samples stored by the class can be defined in the game object containing this script, as seen in **Figure 52**, by changing the value of the variable *Number Of Samples To Store*.

Figure 52: The C# script applied to a game object.

Finally, regarding the inner workings of the base class, all the samples collected are used to provide the rules system functionalities. By using them, the script is able to provide the game designer and developer with values related to the affective state,[195] arousal,[196] valence,[196] sensors,[197] game events[198] and time.[199]

4.3.4 Simplified notation language and structure

The simplified notation language is in fact an XML document structured according to certain rules, which we tried to make as simply as we could. In **Figure 53** we present a template for the XML document.

```
<RULES>
    <RULE action=”action name”>
        <OPERATION leftOp= ”(1)” operator= ”(2)” rightOp= ”(3)”  />
    </RULE>
    <RULE action=”action name”>
        <OPERATION leftOp= ”sensor” leftID=”(ID)” operator= ”(2)”
                   rightOp= ”sensor” rightID=”(ID)”  />
    </RULE>
</RULES>
```

Figure 53: A XML template with structure and syntax necessary for creating rules.

As can be seen in the example, the document starts with a *Rules* tag (<*RULES*>) and end with another (</RULES>). This type of tag denotes the beginning and the end of the area to be parsed by the rules system and, as such, only one should exist.

Inside of the *Rules* tag we find a *Rule* tag (<RULE>). As the name entails, it is here the game designer will specify an individual game rule, using the *Operation* tag. Each rule must contain an *action* attribute: it represents the consequence of this rule and its value will be used by the code generator to name the function generated. For example, *action="makePlayerJump"*

[195] The script is able to determine the current, previous, most common and least common affective state contained in the samples.
[196] The script is able to determine the current, previous, average, highest value and lowest value of this element.
[197] The script is able to determine the current, previous, average, highest and lowest value of each sensor.
[198] The script is able to identify the most recent and previous game event.
[199] This time can either be absolute or relative. See section **3.2.1** for more details on these.

would create the function *public void makePlayerJump()*. *Actions* can be repeated, which means that different rules can trigger the same consequence.

The *Operation* tag can be seen as an *if* condition. If all the conditions of a rule are true, then its action is triggered. An *Operation* must always contain a left operand (represented as *leftOp*), an operator and a right operand (represented as *rightOp*) attribute. These attributes must have a concrete values and cannot be empty, otherwise a parsing error will be thrown. The values each attribute can have are defined in table **Table 10**.

Table 10: Rules system XML file attributes and respective possible values.

Attribute	Values
leftOp	*current_emotion*, *previous_emotion*, *most_common_emotion*, *least_common_emotion*
	current_arousal, *previous_arousal*, *average_arousal*, *highest_arousal*, *lowest_arousal*
	current_valence, *previous_valence*, *average_valence*, *highest_*valence, *lowest_valence*
	sensor_current_value, *sensor_previous_value*, *sensor_average_value*, *sensor_highest_value*, *sensor_lowest_value*
	current_event, *previous_event*
	absolute_time, *relative_time*
operator	*higher*, *lower*, *higherequal*, *lowerequal*, *equal*, *different*[200]
rightOp	Same values as ***leftOp***, text or numbers

As the reader might have noticed, this values are directly linked to the functionalities provided by the base class *BiometricsRuleSet*, described in section **4.3.3**.

If the *leftOp* or *rightOp* are assigned sensors related values, then each must specify the sensor ID. This is done by adding the *leftID* attribute and/or *rightID* attribute to the operation tag. For example, the rule *if the sensor current value, with the ID 'sensor_example_1', equals 0.8 then do avatarAttacks* could be represented as seen in **Figure 54**.

[200] Each value symbolizes either a relational (>, <, >=, <=) or an equality (==, !=) operator.

```
<RULES>
    <RULE action=”avatarAttacks”>
        <OPERATION leftOp=”sensor_current_value” leftID=””
                   operator= ”equal” rightOp= ”0.8”  />
    </RULE>
</RULES>
```

Figure 54: Example of a rule using a sensor.

Finally, it also possible to concatenate several operations inside one rule. Take the example present in **Figure 55**. This example could be read as: *if the arousal average is higher or equal than 0.4 AND if the valence average is higher or equal than 0.4 then do action7*.

```
<RULES>
    <RULE action=”action7”>
        <OPERATION leftOp=”average_arousal” operator=”higherequal”
                   rightOp=”0.4”  />
        <OPERATION leftOp=”average_valence” operator=”higherequal”
                   rightOp=”0.4”  />
    </RULE>
</RULES>
```

Figure 55: Concatenation of operations inside a rule.

With this, all rules have been detailed. We expect that this system will allow the game designer to tailor sufficient complex game mechanics as well reduce the amount of time lost by the game developer when implementing the game mechanics in the game engine. However, the rules system is far from perfect as it only offers a restrictive set of values. For example, if the game designers wished to use the standard deviation as a value for a rule, it would not be possible.

Feedback from game designers and developers would be required in order to improve and make this system even more valuable. Nevertheless, as a prototype, we believe this system as able to implement the profiling tool discussed in section **2.5** and **3.2.2.3** successfully.

4.4 Summary and conclusions

In this chapter we presented the implementation of the framework which as conceptualized and detailed in **Conceiving a** framework for the manipulation of video game elements using the player's biometric data This framework took shape of two components: one named Biometrics Input, which is external to the game and responsible for collecting physiological data sent by the sensors, generate an arousal and valence value, determine the affective state of the player and send all this information to the component inside the game; and other named Biometrics Core, an element of the framework contained inside the game engine, responsible for unpacking the information sent by the previous component and translate into inputs understandable by the game system.

We also introduced our implementation of the rules system, a system which allowed the game designer or developer to define a set of rules, using a simplified notation language, that would then be transformed into code that could be readily in game engine.

Chapter 5

Testing the framework on a simple game prototype

In this chapter we present the small game prototype developed to show the potential of the developed framework. As we were also trying to test if it could be installed in a game that did not previously possess biofeedback, we decided that this prototype should consist of an adaptation of an already existing and simple game.

Throughout this chapter we will detail the selected game, the implemented rules using our simplified notation language, the developed game mechanics and their interaction with the rules, the installation of the Biometrics Core in the game and the configuration of the framework as a whole.

5.1 Selecting a game

Our goal was to find a video game design by others, with simple mechanics, so that we were able to test the implementation of our framework and study the results based on a concrete example.

Unity3D website offers a learning corner[201] where users can find tutorials and documentation related to the game engine. Among these, there is a section dedicated to game projects: development of a small games, typically of one level, from the very beginning. These included the necessary resources[202] and are accompanied by a video walkthrough detailing every step of the game development process and the engine functionalities. This proved beneficial to us, as we could study every component and mechanic in a game designed by others, and still understand it deeply enough to implement our framework.

[201] Which can be find at: http://unity3d.com/pt/learn (accessed June 30, 2016).
[202] Sound, images, 3D models, textures, scripts, the completed game, among others.

In the end, we selected *Survival Shooter*,[203] a 3D isometric third-person shooter, where players control a small human like creature in an oversized room and must survive as much time as possible the waves of dolls pursuing them. To survive, the avatar carries gun which can be use to shoot down enemies, receiving points for each one felled. There are three types of foes, varying in size and strength. The game ends when the player runs out of life, resulting from the attacks suffered. A screenshot of the game project van be found at **Figure 56**.

Figure 56: *Survival Shooter* screenshot.

5.2 Game prototype apparatus and software

Regarding the apparatus, for the development and testing of the game prototype, we used:

1. A Samsung Chronos 7 series computer, model 700Z5C-S03,[204] running a Windows 8.1 operative system;
2. Plux's BITalino board kit.[205]

Regarding software:

1. Unity3D game engine, version 5.3.3f1 personal;
2. The Biometrics Input, as the external component of the framework;[206]
3. The Biometrics Core;[207]
4. The Survival Shooter game project;
5. Simulator of a sensor input.[208]

[203] This game project can be download at: http://unity3d.com/pt/learn/tutorials/projects/survival-shooter-tutorial (accessed June 30, 2016).
[204] Specifications can be found at: http://www.saveonlaptops.co.uk/NP700Z5C-S03UK-Samsung-700Z5C_1281891.html (accessed June 30, 2016).
[205] More specifications can be found here: http://www.bitalino.com/index.php/board-kit (accessed June 30, 2016).
[206] The source code and Eclipse project can be found at: https://figshare.com/s/1407363596dea57676cd (accessed June 30, 2016). Both are contained in the ZIP file named *BiometricsInput2.zip*.
[207] This component of the framework is available, as a Unity package, at: https://figshare.com/s/1407363596dea57676cd (accessed June 30, 2016).
[208] Included in the Eclipse project, together with Biometrics Input.

It is also worth mentioning that final game prototype, with the game mechanics and framework, is already available online.[209]

5.3 Creating rules

When creating the rules, our objective was to use at least one value provided by each layer of the framework[210] to influence the game. Inspired by the categories presented in section **3.1**, we created four rules.

The first was focused on altering the avatar's movement speed based on the player's current[211] valence: the more positive the emotion the player experienced the faster the avatar moved, with the opposite also being true.

The second controls the strength of the enemies based on the player's arousal. This way, the more stressed the player is, the greater the damage suffered when in contact with the enemy. A screenshot of the player fighting and running from the enemies is shown at **Figure 57**.

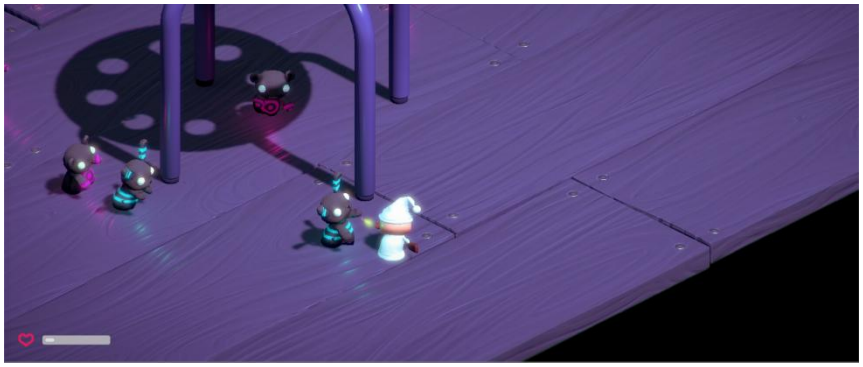


Figure 57: The player, fighting the enemies, while avoiding contact.

The third rule focuses on the aesthetics of the game world, using the player's affective state. In here, we assumed we would be using a circumplex model of affect similar to the one presented in section **2.2.1** (**Figure 3**): four affective states[212] plus a neutral one. As such, when the player is happy or enraged, the mood of the game would become brightly and funny, with an upbeat soundtrack in the background, as they are not spooked with the environment. If, however, they are scared or calm then the environment becomes dark, with a sinister soundtrack. These changes in mood can be observed in **Figure 58**.

[209] Downloadable at the: https://figshare.com/s/1407363596dea5767 6cd (accessed June 30, 2016), available as a ZIP file.
[210] Namely, affective state (provided by the third level), arousal or valence (provided by the second) and the sensor values.
[211] By current, we mean the most recent value produced by the arousal and valence generator.
[212] Happiness, sadness, calm and rage.

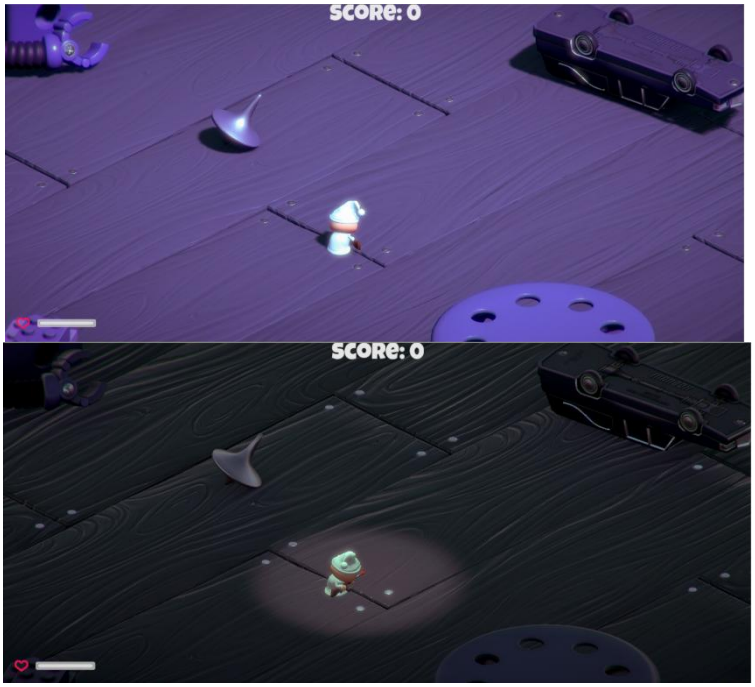


Figure 58: An upbeat and bright mood (above) versus a dark and creepier environment (bellow).

The final rule focuses on the sensor data. We decided to use one sensor and map the values it transmitted directly to the shooting game mechanic. This way, when the biometric data received exceeded a certain threshold, the avatar attacks the enemies. This, however, is just a complement to the input, as the player still needs to use the arrow keys to move his character and the mouse to aim the gun. This sensor was given the ID of *ShootingSensor*. The shooting animation is observable in **Figure 59**.

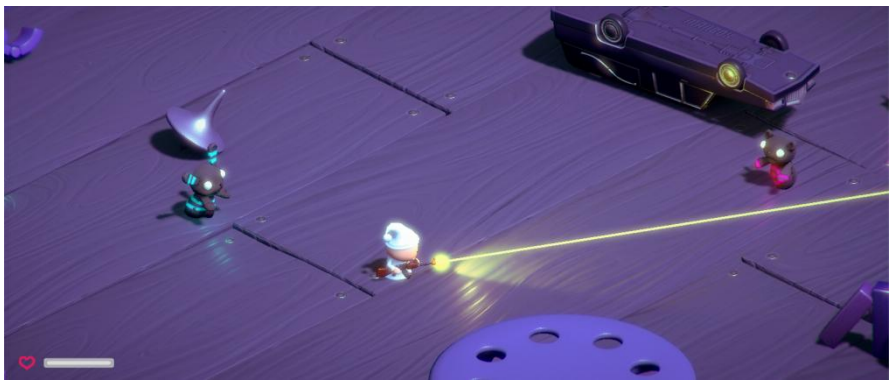


Figure 59: The shooting animation.

The resulting rules are presented bellow, written according to the simplified notation language developed in this work.

```
<?xml version="1.0" encoding="UTF-8"?>
<RULES>
    <!-- Depending on the player's valence, alter the avatar speed -->
    <RULE action="avatarSpeedLow">
      <OPERATION leftOp="current_valence" sign="lower" rightOp="-0.4"/>
    </RULE>
    <RULE action="avatarSpeedNormal">
      <OPERATION leftOp="current_valence" sign="higherequal" rightOp="-0.4"/>
      <OPERATION leftOp="current_valence" sign="lowerequal" rightOp="0.4"/>
    </RULE>
    <RULE action="avatarSpeedHigh">
      <OPERATION leftOp="current_valence" sign="higher" rightOp="0.4"/>
    </RULE>

    <!-- Depending on the player's arousal, change enemies damage -->
    <RULE action="strongEnemies">
      <OPERATION leftOp="current_arousal" sign="higherequal" rightOp="0.8"/>
    </RULE>
    <RULE action="normalEnemies">
      <OPERATION leftOp="current_arousal" sign="higherequal" rightOp="0"/>
      <OPERATION leftOp="current_arousal" sign="lower" rightOp="0.8"/>
    </RULE>
    <RULE action="weakEnemies">
      <OPERATION leftOp="current_arousal" sign="lower" rightOp="0"/>
    </RULE>

    <!-- Depending on the current emotion, change the level atmosphere -->
    <RULE action="setLevelMoodToDark">
      <OPERATION leftOp="current_emotion" sign="equal" rightOp="Sadness"/>
    </RULE>
    <RULE action="setLevelMoodToDark">
      <OPERATION leftOp="current_emotion" sign="equal" rightOp="Calm"/>
    </RULE>

    <RULE action="setLevelMoodToFunny">
      <OPERATION leftOp="current_emotion" sign="equal" rightOp="Happiness"/>
    </RULE>
    <RULE action="setLevelMoodToDark">
      <OPERATION leftOp="current_emotion" sign="equal" rightOp="Rage"/>
    </RULE>

    <!-- Sensor triggers the shooting action -->
    <RULE action="shootEnemies">
      <OPERATION leftOp="sensor_current_value" leftID="ShootingSensor"
                 sign="higherequal" rightOp="750"/>
    </RULE>
</RULES>
```

Unfortunately, rules that touched on themes related to dialogue, narrative or game level generation could not be applied in this game as it did not provide the resources to readily do so.[213] However, at the framework's current state of development, we are now able to develop a more complex game prototype to test these particular subjects, further improving our work in future studies.

[213] Technically, the resources provided could be expanded possibly allowing the implementation of the said topics. However, this would take time and might leave the scope of this dissertation.

5.4 Configuring the framework

To test and developed the game prototype, one sensor must be registered (the *Shooting Sensor*), as demanded by the rules presented before. To add it to the framework, we use the *Register Sensor* graphical interface (found in Biometrics Input, sensors menu). Once registered, the stored values should match the ones presented in **Figure 60**.[214]

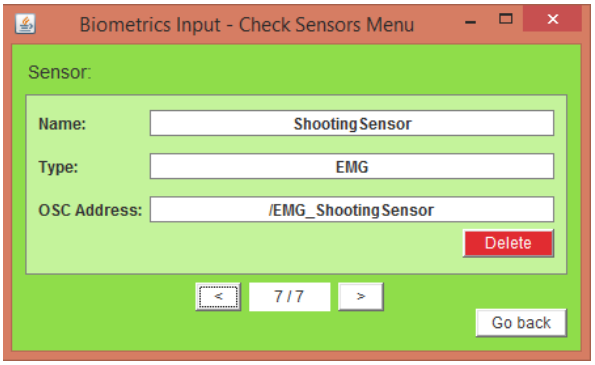


Figure 60: *ShootingSensor* data.

Regarding the circumplex model of affect, the one used contains five emotions: happiness, sadness, calm, rage and neutral; and is not a perfect circle (having its valence dimension smaller than the arousal).[215] When uploaded to the system, the stored information should match **Figure 61**.

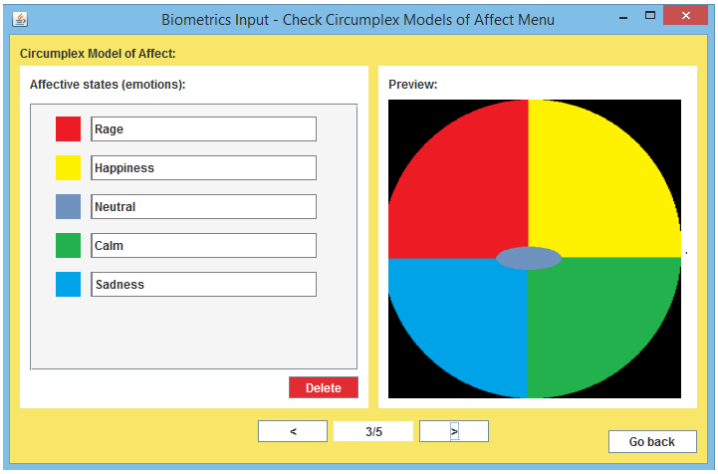


Figure 61: Data of the circumplex model of affect used.

Finally, regarding the Biometrics Input configuration, we performed the test with both the game and framework in the same machine, using the settings present in **Figure 62**. We defined port 8050 as the listening port of Biometrics Input and port 8100 as the listening port of

[214] We considered this sensor to be an EMG.
[215] This model is also available at: https://figshare.com/s/1407363596dea57676cd (accessed June 30, 2016). The file is named 4.png.

Biometrics Core. *Listen IP address* is specified automatically by the framework and *game IP address* was set to 127.0.0.1.[216]

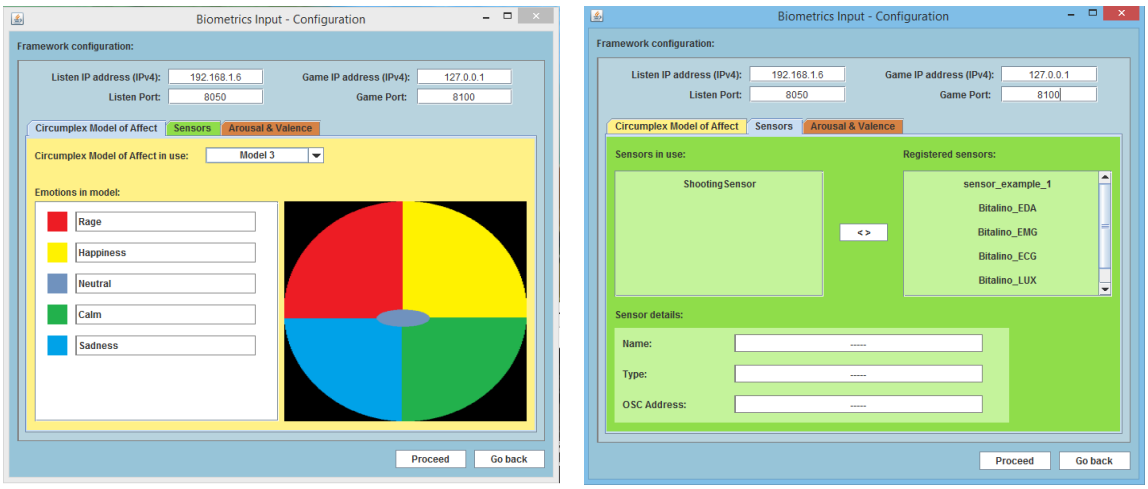


Figure 62: Biometrics Input configuration for the game prototype.

At a later date, we also performed a similar test but with the game and framework running in different machines. Unfortunately, they were unable to communicate with each other. We believe this might be the result of the network configuration or the action of a firewall but, unfortunately, we were unable to confirm these facts.

5.5 Installing Biometrics Core in the game

To install Biometrics Core, we opened the game project and followed the process described in section **4.2.5**. With all the framework files imported, we dragged *Biometrics prefab* into the hierarchy and made sure that the configurations matched the ones set in Biometrics Input, as can be seen in **Figure 63**.

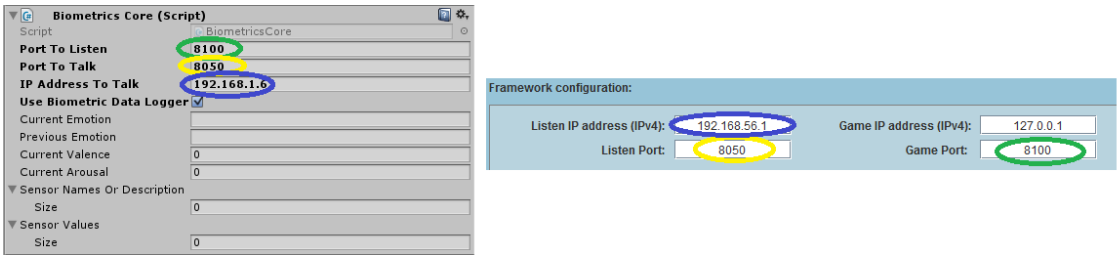


Figure 63: Biometrics Input and Biometrics Core settings must match.

In this example, we decided to use the logger,[217] although it was not mandatory.

[216] 127.0.0.1 represents the local host (in other words: *this machine*).
[217] We checked the 'Use Biometric Data Logger' box. This allowed us to collect several examples, with one of them present in **Biometrics Logger excerpt**.

5.6 Importing the game rules

In order to import the rules defined by us in section **5.3** we used the rules generation functionality provided by Biometrics Input. To do this, we feed it the XML document containing the rules, set its name to *GamePrototype*, and imported the resulting file into the game project. The process is exemplified in **Figure 64**.

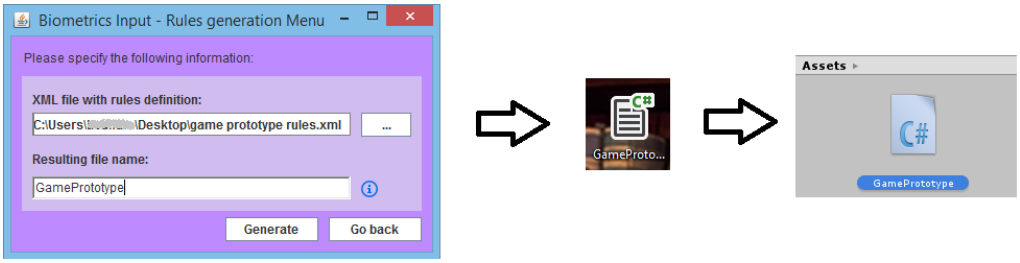


Figure 64: Game rules process creation and import process.

5.7 Changes to the game prototype

With biofeedback now possible, we need to perform the necessary changes in the original game so it can be influenced by the rules created. These changes are detailed bellow:

1. A source of light, positioned above the player, was added to the game hierarchy.

This can be seen in **Figure 65** and was done with the intent of causing a change in the game environment.[218] This change would be triggered by the one of defined rules. However, we could argue that this change did not result from the inclusion of the biofeedback but as a result of a mechanic that demanded functionalities that the game did not possess, and still could be used even without biofeedback.

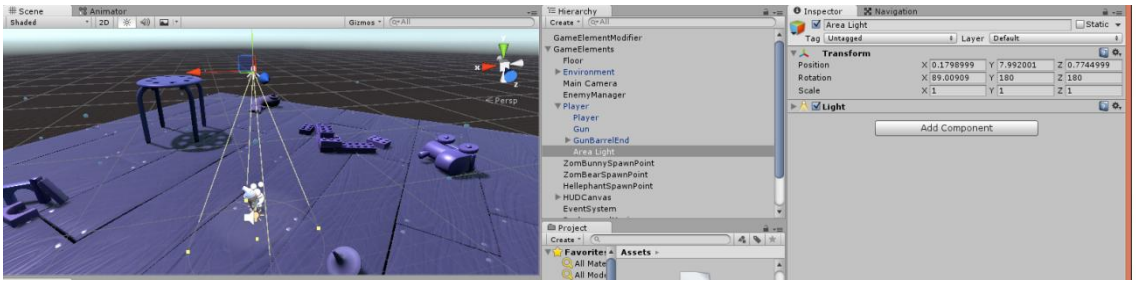


Figure 65: Changes in the game hierarchy.

2. The script *EnemyAttack* was edited.

[218] We would turn off all the game lightning, leaving only the this specific light. The result can be observed in **Figure 58.**

This edition resulted from the addition of a static variable, named *extraDamageArousal*, that stores a integer value representing the extra damage the enemy deal the player depending on the arousal. The changes in the script are presented bellow.

```
public static int extraDamageArousal = 0; // The new variable <<----

void Attack ()
{
      // Reset the timer.
      timer = 0f;

      // If the player has health to lose...
      if(playerHealth.currentHealth > 0)
      {
            // ... damage the player.
            playerHealth.TakeDamage ( attackDamage +
                  extraDamageArousal); // Added here <<----
      }
}
```

3. The *PlayerShooting* script was edited.

This edition was caused due to how the shooting mechanic was implemented. Originally, when the correct input was performed, the game would check if the necessary amount of time between shoots had passed and if the game was not paused, only then it would trigger the *Shoot()* function, which would be perform the necessary actions. However, we wanted to map the *ShootingSensor* directly to this mechanic, which was possible but would create a problem: whenever the value received for this sensor surpassed a certain threshold, the function would be immediately activated , ignoring the time interval or even if the game was paused. To avoid this situation, changes had to performed.

The original and edited script are presented bellow.

```
Original version:

void Update ()
{
      ...
      if( Input.GetButton ("Fire1") && timer >= timeBetweenBullets &&
          Time.timeScale != 0)
      {
      // ... shoot the gun.
            Shoot ();
      }
      ...
}
public void Shoot ()
{
      // Reset the timer.
      timer = 0f;

      // Play the gun shot audioclip.
      gunAudio.Play();
            ...
}
```

```
Edited version:

void Update ()
{
        ...
        if( Input.GetButton ("Fire1") )
        {
        // ... shoot the gun.
              Shoot ();
        }
        ...
}

public void Shoot ()
{
        if (timer >= timeBetweenBullets && Time.timeScale != 0)
        {
              // Reset the timer.
              timer = 0f;

              // Play the gun shot audioclip.
              gunAudio.Play();
        ...
}
```

5.8 Implementing the mechanics

We decided to implement the mechanics separately from the C# rules script so it would be easier to debug. This way, if one mechanic was not having the expected behaviour, we would immediately know it was wrongly implemented.

To do this a created a new script, named *GameElementsModifier*, that received as an input every element to be influenced and contained nine functions able to alter the game. These are detailed and explained in **Table 11**.

Table 11: Functions responsible for influencing the game elements.

Functions	Explanation
alterPlayersPerformanceLow()	These three functions focus on altering the avatar's movement speed. *Normal* is the default value at the start of the game while *Low* and *High decrease* and *increase*, respectively, this value.
alterPlayersPerformanceNormal()	
alterPlayersPerformanceHigh()	
makeEnemiesStronger()	Similar to the three previous functions, these will alter the enemy damage done to the player. This is achieved by altering the static variable *extraDamageArousal* to desired value.
makeEnemiesNormal()	
makeEnemiesWeak()	
changeAestheticsPositiveEmotion()	These functions are responsible for altering the game's atmosphere, transforming it into a bright or spooky

changeAestheticsNegaticeEmotion()	environment, respectively.
shootThroughBiosignal()	This function is responsible for triggering the shooting mechanic.

This script was also created in a way that is possible to trigger the previous function by using the game object editor during the game, as seen in **Figure 66**.

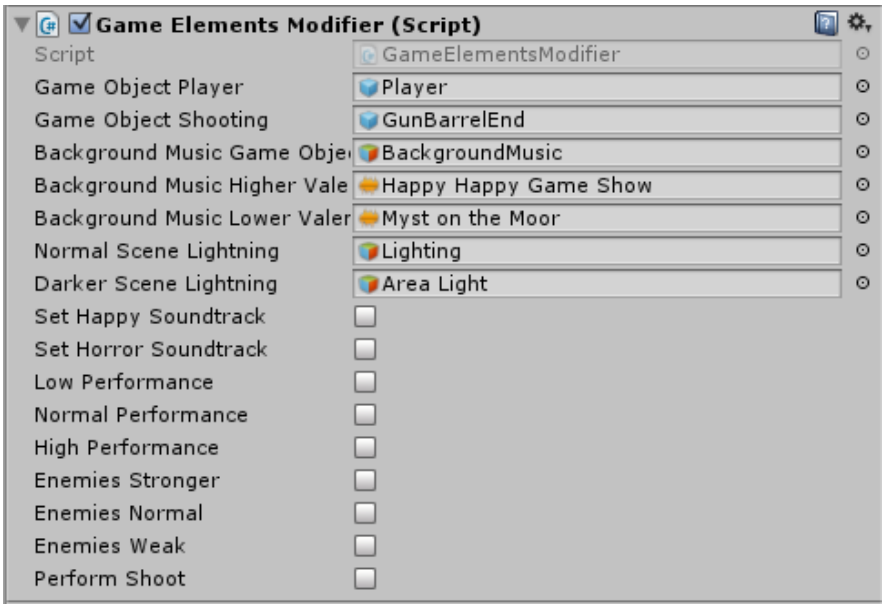


Figure 66: *GameElementsModifier* script.[219]

By adding this script[220] to an game object in the hierarchy (we used an empty object) we could test the functions behaviour. To do this, we would check one of the boxes and the respective functions would be triggered.[221]

5.9 Linking the mechanics with the rules

With the rules already imported into the game engine and the mechanics working correctly, it is now time to join both together in a single script. However, wanting to avoid losing the test capabilities provided by the *GameElementsModifier*, we thought that it would be better to keep both separate.

Achieving this was simple and quickly done.[222] However, this action raised an interesting question: perhaps a possible alternative (or the correct solution) to creating a set of empty

[219] *Set Happy Soundtrack* and *Set Horror Soundtrack* also cause the game environment to change.
[220] A copy of this script is also available at **GameElementsModifier script**.
[221] The variables located in the upper part are the game objects this script is able to alter as well as the soundtrack it will use in the specific environment.
[222] The resulting script can be consulted at **GamePrototype script**.

functions at the end of the script, which the developer would fill, would be to state in the XML document which script contained all the actions defined in the rules and trigger these from there. One other point supporting this decision would be the necessity to redo the rules. For example, imagine that a set of rules is established, generated and imported into the game, with the developer implementing these in the same script. However, at a point in time, one or more rules needed to be changed by some reason. This would involve going back to the beginning and rewrite, generate, import and implement everything again. While generating and importing are not troublesome tasks,[223] the reimplementation might be. By having an external script containing these actions, we would avoid the last problem.

Unfortunately, we did not possess the necessary time to test this hypotheses and we will consider as a possible future work.

5.10 Testing the results

With everything set and in order, it is now time to test the framework using the game as a medium. Regarding the original game, we did not perform any tests on it as we assumed it would be working without errors and the process used to test the game mechanics was already described in section **5.8**.

Focusing in the rules, we used the Biometrics Input valance and arousal generator to perform the tests as the graphical interface allowed us to control the values sent to the game and, as consequence, control which rules to trigger. To test the particular rule that used the sensor values as a trigger, we performed two tests, the first using a fake sensor[224] and second one using the BITalino EMG sensor.

While testing, however, we found one structural error that may force us to review how the framework is implemented. The Unity3D game engine offers its developers the possibility of reloading a game level whenever necessary. What this does is delete every game object created in the game hierarchy and recreate them again, and it is here that the problem arises. The Biometrics Core resorts to an auxiliary class to handle the OSC communication. Unfortunately, the resources used by this class are not being freed correctly when the game restarts but only when the game is terminated. As such, when Biometrics Core tries to re-establish the OSC communication using the same port, this port is already occupied, making it unable to proceed.

To solve this problem we would have to study the auxiliary class in more detail to understand its inner workings and, from there, try to determine a solution to this problem. However, we lacked the necessary time to do so, so this task will have to be left to another iteration of the framework.

[223] The rewriting, troublesome or not, would have to be done has there were rules that needed to be changed.
[224] A piece of software that would send random values to the OSC address associated with the *Shooting Sensor*.

Nevertheless, so long as a level is not restarted, the Biometrics Core proved to be able to perform its intended objectives.

5.11 Summary and conclusions

To showcase our framework potentials and try to determine possible shortcomings existing in it, we applied it to a simple game prototype. One important condition was that this game had to developed by others so we could study how difficult was the installation of our framework and how much the game would have to change in order to be compatible with it.

Throughout this chapter we detail the game selected, how the installation process was done, changes that had to be performed in the original game, implemented rules and game mechanics and tests performed. The framework configuration, hardware and software used are also mentioned.

Chapter 6

Conclusions and Future Work

With this dissertation we initiated the development of a framework capable of changing diverse elements of a video game using the player's biometric data. This framework consists of two components: *Biometrics Input* and *Biometrics Core*, each with a very specific function. *Biometrics Input* is the component external to the game, responsible for collecting the players' biometric data and affective state, information which it would send to its counterpart. The *Biometrics Core*, installed inside the game, would receive this information and transform it into data understandable and usable by the game system. This way, an affective and biofeedback system was implemented.

We also developed a rules system, in order to increase the game designer's and game developer's performance when using our framework. This system offers a simplified notation language, with its roots in the eXtensible markup language, allowing the game designer to create rules that directly use the framework 's values as an input. A code generator is also included with system, capable of converting the defined rules into a C# script, immediately compatible with Unity3D game engine. This allows game developers to focus their time in developing the game's mechanics, without having to worry with the setup necessary to access the framework's functionalities.

With the framework completed, we intended to develop a small video game prototype, using our framework as support. To do this, we researched what kind of game mechanics were used in affective and biofeedback video games, either commercial and in scientific experiments; as well as which types of sensors were available in the market, so we could select a set to use in this prototype. This prototype would provide us with a tool for debugging the framework and to find improvements in the installation process.

This game prototype would be followed by an experiment, where game designers and game developers would be invited to test our framework in their projects to determine if it was

indeed a valuable asset or a troublesome component. This would also allow us to receive feedback for new functionalities and/or improvements to implement in future iterations of the framework.

6.1 Limitations

Unfortunately, we were met with some setbacks during the project, making it impossible to achieve all our objectives.

The first problem we ran into was in the implementation of the framework. One of the core components, responsible for reading the player's biometric data and for extract the player's arousal and valence. We tried implementing the PIERS sub-system,[225] developed by Nogueira, into our framework but without success, as it required specific mathematical knowledge that we did not possess. Of course, there was always the possibility of performing a deep researched about the subject but this would consume time, a resource we were already sparing with. To try to tackle the issue, we tried to contact the author directly. Unfortunately, he could not help as he was busy at the time.

To solve this issue, we created a valence and arousal generator that would produce values that smoothly varied over time. This allowed us to finish the framework, but at the price of losing its affective component. This would also render the framework inappropriate for the experiment with the game developers and game designers. However, the framework inner workings remains the same, but with the generator occupying the component.

The other problem we ran into was in the game prototype development, specifically in the selection of sensors. Many of the higher quality sensors researched are expensive and, as such, we add to opt for a lower budget solution. We ended up picking Plux BITalino[226] as our experimental sensor as it offered a set of biometric sensors (EMG, EDA and ECG) in a single package and as available for request at our faculty. However, when testing it, several issues came up: the electrode connectors[227] were too short, making it very hard to use more than one type of sensor at the same time; and the device was easily affected by noise. For example, if we performed an EMG to the hand muscles, it would still be possible to perform an GSR in the same area. However, to perform an ECG we could either: use the same area or choose another. In the former, we risk having our data polluted with noise due to the overlapping connectors.[228] In the last, we would have to pick the chest area, which as beyond the reach of the electrode cables.

We tried to solve these issues issue by extending the electrode cables. This way, we could place the sensors in distinct body parts without risking one crossing each other, has it happened

[225] Nogueira's framework and the PIERS system are described in section **2.4**.
[226] See **Collection of biometric sensors** for more information on this sensor.
[227] The cables that connected the sensor with its electrode.
[228] It is also worth mentioning that it is challenge to place all the electrodes in such a small area.

when they were all crumpled together in the same area. Unfortunately, it turned out the materials and tools to do this were expensive, forcing us to discard this idea.

Another problem faced during the game development as the incapacity of the Biometrics Core to deal with the restarting of the game level, as done by the Unity3D game engine, causing the OSC communication to become unavailable, severely restricting the framework functionalities. While we managed to sidestep this problem, and there are ways of dealing with this situation, removing this issue should be one of top objective in the next framework iteration.

Although not as major as the previous limitations, some visual bugs where found when debugging the framework. None of these prevent the normal flow of the application and can be simply corrected by returning to the previous menu. However, as these were irregular, it was not possible to determine their cause, preventing us from fixing them.

6.2 Future work

Our suggestions for future work come as a direct consequence of the limitations presented in the last section and unmet objectives.

One of these is to implement the framework component responsible for extracting the player's arousal and valence. The process on how to do so is already detailed in this work, in section **2.4**. All that remains his acquiring the mathematical knowledge on how to do so and implement it. With this it would be possible to develop a fully working framework enabling the affective and biofeedback in the video game using it. As a consequence, we would also be able to perform the experiment with game designers and developers as well as develop a more powerful and complex prototype showcasing the framework possibilities.

Even without the improvement described before, several actions remain to us that could not be completed in this work, namely the creation of a game prototype, more vast in terms of the offered mechanics, so that different and more complex rules could be implemented and their results studied.

It is also our belief that, if the Biometrics Core was slightly altered, it would prove compatible with other systems besides the Unity3D game engine. This way, it could be applied to other areas besides video games, such as the study of the User Experience or medical application that use biofeedback as a way to interact with their patients.

Finally, we see that this framework will allow us to pursue our initial idea: exploring the relationship between biofeedback and narrative in games, studying how the player's biometric data may shape narrative, providing improved and more personalised experiences. Also, developing a prototype more complex in nature

Bibliography

Aguiar, Rúben Pinto. "Affective Game Design: Creating Better Game Experiences based on Players Affective Reaction Model." Dissertation, 2014.

Ambinder, Mike. "Biofeedback in Gameplay: How Valve Measures Physiology to Enhance Gaming Experience." *Valve.* March 3, 2011. http://www.valvesoftware.com/publications/2011/ValveBiofeedback-Ambinder.pdf (accessed May 19, 2015).

Batson, C. Daniel, Laura L. Shaw, and Kathryn C. Oleson. "Differentiating affect, mood, and emotion: Toward functionally based conceptual distinctions." *Emotion. Review of personality and social psychology.* Thousand Oaks, CA, US: Sage Publications, 1992. 294 -326.

Bersak, D., et al. "Intelligent Biofeedback using an Immersive Competitive Environment." *Online Proceedings for the Designing Ubiquitous Computing Games Workshop.* 2001.

Cardoso, Pedro. "Playing in 7D: An Action-Oriented Framework for Video Games." Ph.D. Thesis, Faculty of Fine Arts, University of Porto, 2015.

Dekker, Andrew, and Erik Champion. "Please Biofeed the Zombies: Enhancing the Gameplay and Display of a Horror Game Using Biofeedback." *Situated Play, Proceedings of DiGRA 2007 Conference* , 2007: 550-558.

Droog, Simon. "What is emotion? (part 1) – 4 Affective states." *Experiencing Architecture.* 24 January 2010. https://experiencingarchitecture.com/2010/01/24/what-is-emotion-part-1/ (accessed June 9, 2016).

—. *What is emotion? (part 3) – 4 Ways of Manifestation.* 31 January 2010. https://experiencingarchitecture.com/2010/01/31/what-is-emotion-part-3-4-ways-of-manifestation/ (accessed Junho 9, 2016).

Ekkekakis, Panteleimon. "The measurement of affect, mood, and emotion in exercise psychology." Edited by G. Tenenbaum, R.C. Eklund and A. Kamata. *Measurement in sport and exercise psychology*, 2012: 321-332.

Ekman, Paul. "Basic Emotions." *The Handbook of Cognition and Emotion*, 1999: 45-60.

Gilleade, Kiel Mark, Alan Dix, and Jen Allanson. "Affective Videogames and Modes of Affective Gaming: Assist Me, Challenge Me, Emote Me." *DiGRA 2005: Changing Views – Worlds in Play.*, 2005.

Kuikkaniemi, Kai and Laitinen, Toni and Turpeinen, Marko and Saari, Timo and Kosunen, Ilkka and Ravaja, Niklas. "The Influence of Implicit and Explicit Biofeedback." In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 859 - 868. New York, NY, USA: ACM, 2010.

LeBlanc, Marc. "Tools for Creating Dramatic Game Dynamics." *The Game Design Reader—A Rules of Play Anthology, edited by Eric Zimmerman Katie Salen*, 2005: 438 - 459.

Lövheim, Hugo. "A new three-dimensional model for emotions and monoamine neurotransmitters." *Medical Hypotheses* 78 (2011): 341 - 348.

Mirza-Babaei, Pejman, Sebastian Long, Emma Foley, and Graham McAllister. "Understanding the Contribution of Biometrics to Games User Research." *Proceedings of DiGRA 2011 Conference: Think Design Play.*, 2011.

Nacke, Lennart, Michael Kalyn, Calvin Lough, and Regan Mandryk. "Biofeedback Game Design: Using Direct and Indirect Physiological Control to Enhance Game Interaction." *Proceedings of the 2011 Annual Conference on Human Factors in Computing Systems*, 2011: 103-112.

Nóbrega, Artur. "Geração Procedimental Baseada em Dados Afectivos." Master thesis, 2014.

Nogueira, Pedro A., Rui Rodrigues, Eugénio Oliveira, and Lennart E. Nacke. "Guided emotional state regulation: Understanding and shaping players' affective experiences in digital games." *Association for the Advancement of Artificial Intelligence*, 2013: 51 - 57.

Picard, Rosalind W. "Affective Computing for HCI." *Proceedings of HCI International (the 8th International Conference on Human-Computer Interaction)*, 1999: 829 - 833.

Plutchik, Robert. "The Nature of Emotions." *American Scientist* 89 (July - August 2001): 344 - 350.

Posner, Jonathan, James A. Russell, and Bradley S. Peterson. "The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology." *Development and Psychopathology*, 2005: 715 - 734.

Rego, David de Oliveira. *Biofeedback Augmented Gameplay: Criação de Uma Framework para o Desenvolvimento de Mecanismos de Interação Fisiológica em Videojogos Comerciais.* Dissertation, Porto, Portugal: Faculdade de Engenharia da Universidade do Porto, 2014.

Reynolds, Erin Elizabeth. ""Nevermind: Creating an Entertaining Biofeedback-enhanced Game Experience to Train Users in Stress Management." Thesis for the Master of Fine Arts (Interactive Media), 2012.

Russell, James .A, and L. Feldman Barrett. "Core affect, prototypical emotional episodes, and other things called emotion: Dissecting the elephant." *Journal of Personality and Social Psychology, 76*, 1999: 805-819.

Russell, James .A, and L. Feldman Barrett. "Core affect." Edited by D. Sander and K. R. Scherer. *The Oxford companion to emotion*, 2009.

Russell, James A. "A circumplex Model of Affect." *Journal of Personality and Social Psychology*, 1980: 1161-1178.

Silva, Gonçalo Filipe Lopes Coelho Amaral da. "Multimodal vs. Unimodal Physiological Control in Videogames for Enhanced Realism and Depth." Master Thesis, 2014.

Torres, Vasco Pereira. "Development of Biofeedback Mechanisms in a Procedural Environment Using Biometric Sensors." Master Thesis Dissertation, 2013.

Zagalo, Nelson. *Emoções interactivas. Do cinema para os videojogos.* Grácio Editor, 2009.

Cited works

Atelier Shallie. 2015. Gust Co. Ltd., and Koei Tecmo.

Bravely Default. 2013. Silicon Studio, and Square Enix.

Dark Souls. 2012. FromSoftware, and Namco Bandai Games.

Fallout 4. 2015. Bethesda Game Studios, and Bethesda Softworks.

Heavy Rain. 2010. Quantic Dream, and Sony Computer Entertainment.

Keep Talking and Nobody Explodes. 2015. Steel Crate Games.

League of Legends. 2009. Riot Games.

Left 4 Dead. 2008. Turtle Rock Studios, and Valve Corporation.

Left 4 Dead 2. 2009. Valve Corporation.

Metal Gear Solid V: The Phantom Pain. 2015. Kojima Productions, and Konami Digital Entertainment.

Nevermind. 2015. Flying Mollusk.

New Super Mario Bros. U. 2012. Nintendo.

Street Fighter II. 1991. Capcom.

Sid Meier's Civilization III. 2001. Fireaxis Games.

Silent Hill 2. 2001. Konami Computer Entertainment Tokyo.

Super Mario Bros.. 1985. Nintendo.

Super Mario 3D World. 2013. Nintendo.

Super Metroid. 1994. Intelligent Systems, and Nintendo.

VANISH. 2013. 3DrunkMen.

Addendum A

Collection of biometric sensors

A collection of biometric sensors researched during the development of this project can be found in this addendum. Attention was given so that all the biometric signals described in section **2.3.4** had a sensor able to capture it. Each entry contains information on the sensor, namely its name, type of biometric data able to be collected and a small description. Links with more information about the device and/or purchasing location are also given.

Table A1: Information regarding different sensors

Name	Data collected	Extra information
Affectiva GSR Qsensor	GSR, Temperature	No longer in production.[229]
ASL Mobile Eye-XG[230]	Eye tracking	Automatic calibration. Has two cameras (one captures the eyes of the user, the other what he/she is seeing). Uses a storage media (SD, Micro SD, SD HC Card), up to 32Gb of recording.
Athos apparel[231]	Heart rate, muscle tension (EMG)	Product differs by gender: number of sensors change but functionality remains the same.

[229] As seen here: http://imotions.com/portfolio-item/affectiva-gsr-qsensor/ (accessed January 25, 2016).
[230] Mobile Eye-XG brochure: http://www.asleyetracking.com/site/Portals/0/Subpages/documents/Mobile%20Eye-XG%20Brochure%2060Hz.pdf (accessed January 28, 2016).
[231] Company website: https://www.liveathos.com/products# (accessed January 25, 2016).

B-alert X10 EEG Headset System[232] / B-alert X24 EEG Headset System[233]	Brain activity	Works through Bluetooth. Compatible with PC. Both models are similar, with X24 offering extra channels for EEG.
Basis' Peak[234]	Heart rate	Only compatible with smartphones.
Biometrics' Surface EMG[235]	Muscle tension (EMG)	This sensor requires Biometrics' DataLog[236] (a decoder) to work.
Body Temperature Sensor for e-Health Platform[237]	Temperature	Requires e-Health Sensor Shield V2.0 (the decoder) to work, with this one being connected to ARDUINO, Raspberry Pie or Intel Galileo in order to work.
CardioChip™ ECG biosensor[238]	Stress, heart rate, heart rate variability, fatigue, breathing Index, mood (ECG)	The extracted data seems to result of the calculations performed by an algorithm using the ECG as an input.
e-Health Sensor Platform Complete Kit[239]	Brain activity, Breathing, GSR, Muscle Tension, Oxygen, Temperature	This a set of sensor bundled with a decoder. Must be connected to ARDUINO, Raspberry Pie or Intel Galileo in order to work. Heart rate Also allows to determine the glucose level[240] and blood pressure.
Emotic EPOC / EPOC+[241]	Brain activity, Facial expressions	Cross-platform compatible. Each model offers different sample rates (128 SPS / 256 SPS). Can use Wireless or Bluetooth for communication. Can detect smiles[242].

[232] Sensor specifications can be found at: http://www.advancedbrainmonitoring.com/xseries/x10/ (accessed January 25, 2016).
[233] Sensor specifications can be found at: http://www.advancedbrainmonitoring.com/xseries/x10/ (accessed January 25, 2016).
[234] Sensor webpage can be found at: http://www.mybasis.com/blog/2014/11/basis-peak-fitness-heart-rate-arrived/ (accessed January 25, 2016).
[235] Product specifications webpage: http://www.biometricsltd.com/semg.htm (accessed January 25, 2016).
[236] Biometric's DataLog specifications can be found at: http://www.biometricsltd.com/datalog.htm (accessed January 25, 2016).
[237] Temperature sensor product page: https://www.cooking-hacks.com/body-temperature-sensor-ehealth-medical (accessed January 28, 2016).
[238] CardioChip™ ECG biosensor website: http://neurosky.com/biosensors/ecg-sensor/ (accessed January 25, 2016).
[239] Sensor description and other information can be found at: https://www.cooking-hacks.com/documentation/tutorials/ehealth-biometric-sensor-platform-arduino-raspberry-pi-medical (accessed January 25, 2016).
[240] Gives the concentration of glucose in blood.
[241] Product webpage: https://emotiv.com/epoc.php (accessed January 25, 2016).

Emotiv Insight[243]	Brain activity, Facial expressions	Has an SDK for development. Similar specifications to Emotic EPOC / EPOC+. Is able to measure attention, focus, engagement, interest, excitement, affinity, relaxation and stress levels[244]. Is able to detect certain facial expressions such as: blinks, winks, frown, surprise, clench and smile.
Empatica E4 wristband[245]	GSR, Heart Rate	Has internal storage. Communication done through Bluetooth.
Eye Tribe Tracker PRO[246]	Eye tracking	Sampling rate from 30Hz to 75Hz
EyeTech VT3 mini[247]	Eye tracking	Sample rate of 60Hz. Compatible with Windows 7/8 and Android 32/64-bit. Connection done through USB.
Garmin Soft Strap Premium Heart Rate Monitor[248]	Heart rate	Only compatible with ANT+ (not possible to connect to a computer without an extra component/accessory).[249]
Gazepoint GP3 Eye Tracking[250]	Eye tracking	60Hz update rate. Open standard API.
Imotions' software[251]	Facial expressions, eye tracking	This software is able to use a webcam for capturing facial expressions. A specific hardware is required for eye tracking.[252] It's also compatible with many type of

[242] As seen in page 2 of the specification sheet: https://emotiv.com/product-specs/Emotiv%20EPOC%20Specifications %202014.pdf (accessed January 25, 2016).
[243] Insight specifications and webpage: https://emotiv.com/insight.php (accessed January 25, 2016).
[244] According to the specifications sheet found at: https://emotiv.com/product-specs/Emotiv%20Insight%20Product%20Sheet%202014.pdf (accessed January 25, 2016).
[245] More information can be found at: https://www.empatica.com/e4-wristband (accessed January 25, 2016).
[246] Eye Tribe Tracker Pro technical specifications: https://theeyetribe.com/products/ (accessed January 28, 2016).
[247] VT3 mini product webpage: http://www.eyetechds.com/vt3-mini-research.html (accessed January 28, 2016).
[248] Garmin shop webpage: https://buy.garmin.com/en-US/US/shop-by-accessories/fitness-sensors/soft-strap-premium-heart-rate-monitor/prod15490.html (accessed January 25, 2016).
[249] Here is one possible solution for the connection: http://shop.virtualtraining.eu/en/product/ant-usb-mini-stick/ (accessed January 25, 2016).
[250] GP3 Eye tracker purchase page (with specifications): http://www.gazept.com/product/gazepoint-gp3-eye-tracker/ (accessed January 28, 2016).
[251] Imotions' main webpage can be found at: https://imotions.com/products/ (accessed January 28, 2016).
[252] A list of compatible devices can be found at: http://imotions.com/hardware/remote-eye-trackers/ (accessed January 25, 2016).

		sensors,[253] making the software a central hub for collecting all different kinds of biometric information. Also has algorithms for parsing and analyzing the received data.
MBody BIKE&Run / MBody AllSport [254]	Heart rate, Muscle tension	Wearable shorts with embedded with sensors. Products are similar, differing only in fabric.
Microsoft Kinect 2.0	Posture, Heart rate	The device is able to track the full body in 3d, including wrist rotation and finger tracking, and head tracking. Is able to track a facial expression and recognize a face (however this is different from recognizing different facial expressions in the same person).[255]
Mindwave™ Headset[256]	Brain Activity	Automatic wireless computer pairing, with the device having a MAC address. Works using one AAA battery (6-8 hours runtime).
Mio Link[257]	Heart rate	Uses Bluetooth Smart (4.0) and ANT+ for communication.
myGaze Developer Solution[258]	Eye tracking	There is also an Enterprise, Developer and Academic edition for this product. Includes camera peripheral and software (SDK).
Neurobit Lite™ [259]	Brain activity	Communication with computer done through infrared. The device is a decoder. It uses electrodes to record the signal.
OM Smart Shirt + Smart Box[260]	Heart rate, breathing	Only compatible with iOS. *API* and *SDK* still in development.

[253] A detailed compatibility list can be found at: http://imotions.com/hardware/biometric-sensors/ (accessed January 25, 2016).
[254] Products webpage: http://www.myontec.com/products/ (accessed January 25, 2016).
[255] This information has found at: http://123kinect.com/everything-kinect-2-one-place/43136/ (accessed January 27, 2016).
[256] Mindwave™ Headset website: http://neurosky.com/biosensors/eeg-sensor/biosensors/ (accessed January 25, 2016)
[257] Sensor product webpage can be found at: http://www.mioglobal.com/en-us/Mio-Link-heart-rate-wristband/Product.aspx (accessed January 25, 2016).
[258] myGaze® shop website: http://eu-shop.mygaze.com/developer-solutions/ (accessed January 25, 2016).
[259] Neurobit Lite™ website: http://www.neurobitsystems.com/product.htm (accessed January 25, 2016).
[260] OM Signal website: http://www.omsignal.com/pages/how-it-works (accessed January 25, 2016).

Plux BITalino[261]	Heart Rate, Muscle Tension, GSR	This is not a sensor but a decoder. This device has specifically designed to "learn and prototype applications using body signals". It comes with several sensors, immediately ready to use, and a toolkit that allows the module to be reprogrammed in order to achieve a specific objective. The data collected can vary depending on the model and coponents.
Shimmer3 GSR+ Unit[262]	GSR	It includes pulse sensors for the fingers and earlobe.
Short-Range Intel® *RealSense* Camera™ [263]	Facial expressions	Included in some Laptops. Contains two cameras: 1080HD, infrared and a infrared laser projector.
SMS Audio Biosport[264]	Heart rate	Only compatible with smartphones.
SparkFun Electronics' Muscle Sensor V3[265]	Muscle tension (EMG)	It is a ARDUINO component. Uses electrodes to capture the signal.
SparkFun Pulse Sensor[266]	Heart rate	It is a ARDUINO component.
Thought Technology's EEG Sensor - Model T9305M[267]	Brain activity	This is a decoder. Uses electrodes to capture the signal. Requires an extra componentto work as interface to the computer (uses VGA cable for connection).
Thought Technology's EKG Sensor - Model T9306M / T9307M [268]	Heart rate (ECG)	This is a decoder. Uses electrodes to capture the signal. Requires an extra component to work as interface to the computer (uses VGA cable for connection).
Thought Technology's Respiration Sensor -	Breathing	Requires an extra component[270] to work as a decoder and interface to the computer (uses VGA cable for connection).

[261] BITalino store webpage: http://www.bitalino.com/ (accessed January 25, 2016).
[262] Sensor specifications can be found at: http://www.shimmersensing.com/shop/shimmer3-wireless-gsrsensor#specifications-tab (accessed January 25, 2016).
[263] Intel® *RealSense* Camera Website: http://www.intel.com/content/www/us/en/architecture-and-technology/realsense-shortrange.html (accessed January 25, 2016).
[264] Product webpage at Amazon can be found at: https://www.amazon.com/SMS-Audio-BioSport-Earbud-Black/dp/B00PDHXL8I (accessed January 25, 2016).
[265] Sensor documentation can be found here: https://www.sparkfun.com/products/13027 (accessed January 25, 2016).
[266] Documentation of the sensor can be found at: https://www.sparkfun.com/products/11574 (accessed January 25, 2016).
[267] Thought Technology's EEG Sensor website: http://www.thoughttechnology.com/sciencedivision/product.html (accessed January 25, 2016).
[268] Thought Technology's EKG Sensor website: http://www.thoughttechnology.com/sciencedivision/pages/products/ekg.html (accessed January 25, 2016).

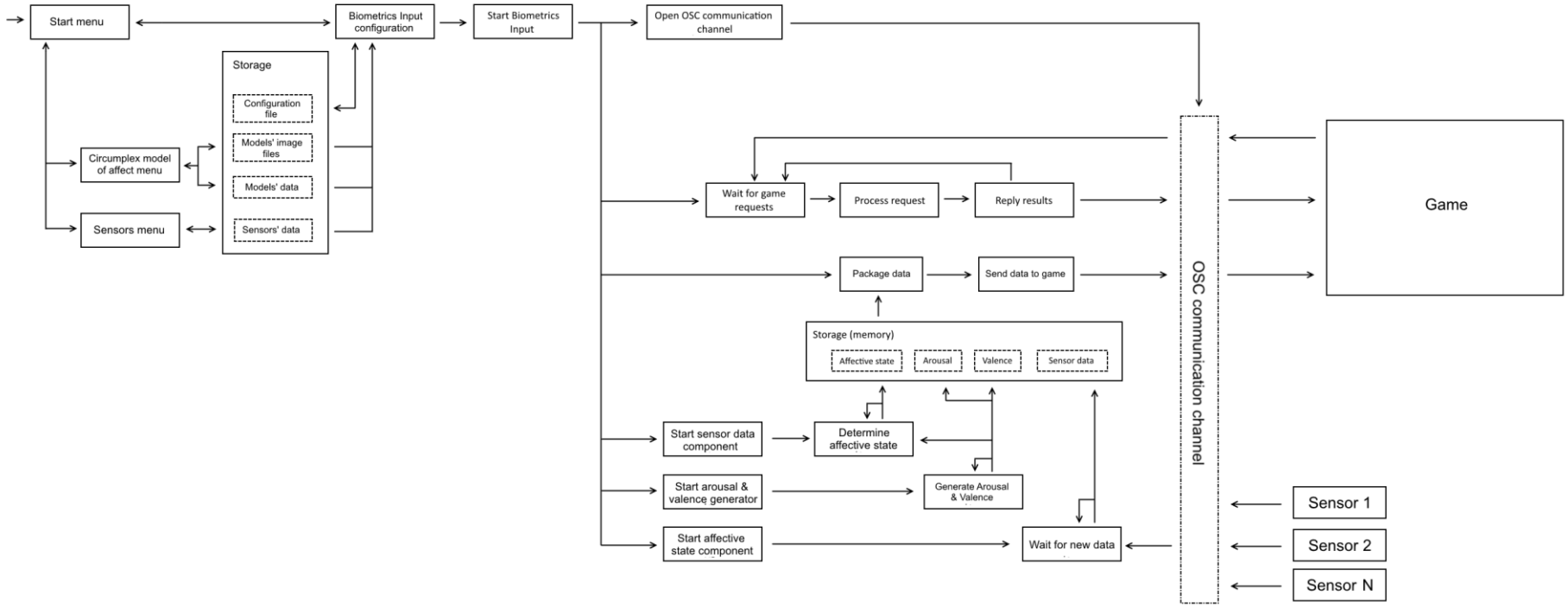
Model SA9311M[269]		
Tobii Pro Glasses 2[271]	Eye tracking	50Hz or 100Hz sampling rate. Has two cameras, one captures what the user is seeing (in HD) the other captures the user gaze. Contains a recording unit, that double as storing unit (SD card), recording image AND sound, and a control unit, used to control the device remotely.
Tobii Pro X3-120[272] **/ X2-60**[273] **/ X2-30 eye tracker**	Eye tracking	Automatic calibration. Cross-platform (works on Windows 7/8.1, Max OS X and Linux), except X2-30 (only compatible with Windows). Different models have different sampling rates (120Hz, 60Hz and 30Hz respectively).
Tobii TX300 Eye Tracker[274]	Eye tracking	Three sampling rates: 60Hz, 120Hz and 300Hz. Automatic calibration. The sensor comes embedded on a monitor, being this connected to the computer/desktop responsible for receiving the data.

Once again, it is important to remind the reader that this collection is limited, not comprising every sensor available on the market.

[270] This component is sold separately at: http://www.thoughttechnology.com/sciencedivision/pages/products/sensoriso.html (accessed January 25, 2016).
[269] Thought Technology's Respiration Sensor website: http://www.thoughttechnology.com/sciencedivision/product.html (accessed January 25, 2016).
[271] Tobii Pro Glasses product description: http://www.tobiipro.com/siteassets/tobii-pro/product-descriptions/tobii-pro-glasses-2-product-description.pdf (accessed January 28, 2016).
[272] Tobii X2-120 specifications: http://www.tobiipro.com/siteassets/tobii-pro/brochures/tobiipro-x3-120-brochure.pdf (accessed January 28, 2016).
[273] Tobii Pro X2-60 and X2-30 specifications: http://www.tobiipro.com/siteassets/tobii-pro/product-descriptions/tobii-pro-x2-product-description.pdf (accessed January 28, 2016).
[274] Tobii TX300 Eye Tracker brochure: http://www.tobiipro.com/siteassets/tobii-pro/brochures/tobii-pro-tx300-brochure.pdf (accessed January 28, 2016).

Addendum B

Framework flow (full diagram)



Addendum C

Biometrics Logger excerpt

The following text is an excerpt of log file produced by Biometrics Logger. The first line represents the header of the file, explaining the meaning of each value. This file could be used as SCV (comma -separated values) file.

```
Time data was collected, Time data was received, Emotion, Valence,
Arousal, Bitalino_ECG, Bitalino_EMG,  Event(s) received
02/07/2016 22:01:05:974, 02/07/2016 10:01:05.975, Rage, -0.26, 0.064,
0, 1,
02/07/2016 22:01:05:990, 02/07/2016 10:01:05.991, Rage, -0.26, 0.06,
0, 1,
02/07/2016 22:01:06:07, 02/07/2016 10:01:06.007, Rage, -0.26, 0.0559,
0, 1,
02/07/2016 22:01:06:23, 02/07/2016 10:01:06.024, Rage, -0.264, 0.0559,
0, 1,
02/07/2016 22:01:06:39, 02/07/2016 10:01:06.039, Rage, -0.264, 0.052,
0, 1,
02/07/2016 22:01:06:55, 02/07/2016 10:01:06.055, Rage, -0.26, 0.0559,
0, 1,
02/07/2016 22:01:06:71, 02/07/2016 10:01:06.071, Rage, -0.256, 0.06,
0, 1,
02/07/2016 22:01:06:87, 02/07/2016 10:01:06.096, Rage, -0.26, 0.06, 0,
1,
02/07/2016 22:01:06:103, 02/07/2016 10:01:06.108, Rage, -0.256, 0.06,
0, 1,
02/07/2016 22:01:06:120, 02/07/2016 10:01:06.120, Rage, -0.252,
0.0559, 0, 1,
```

In this example, *Bitalino_ECG* and *Bitalino_EMG* represented the information received from these sensors. *Event(s) received* are empty as none were received.

Addendum D

Rules system XML file example and resulting code

The following text is an example of XML file using the simplified notation language defined by us.

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- The rules contained in this XML are for testing purposes only-->
<RULES>
    <RULE action="action1">
        <OPERATION leftOp="current_emotion" sign="equal" rightOp="Happy"/>
    </RULE>

    <RULE action="action1">
        <OPERATION leftOp="previous_emotion" sign="different" rightOp="Happy"/>
    </RULE>

    <RULE action="action2">
        <OPERATION leftOp="average_arousal" sign="equal" rightOp="325" />
    </RULE>

    <RULE action="action3">
        <OPERATION leftOp="highest_valence" sign="equal" rightOp="0.235" />
    </RULE>

    <RULE action="action3">
        <OPERATION leftOp="current_valence" sign="higher" rightOp="48" />
    </RULE>

    <RULE action="action5">
        <OPERATION leftOp="current_event" sign="equal" rightOp="Text Here" />
    </RULE>

    <RULE action="action5">
        <OPERATION leftOp="previous_event" sign="different" rightOp="Text here" />
    </RULE>
```

```
    <!-- Example of a concatenation of two operations inside a rule. -->
    <RULE action="action7">
        <OPERATION leftOp="average_arousal" sign="higherequal" rightOp="0.4" />
        <OPERATION leftOp="average_valence" sign="higherequal" rightOp="0.4" />
    </RULE>
</RULES>
```

And the resulting C# script after the parsing and code generation.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Biometrics;

class RulesExampleDissertation : BiometricsRulesetBase
{
    protected override void checkConditionsAndTriggerActions()
    {
        if( getCurrentEmotion().Equals("Happy") )
        {
            action1();
        }

        if( !getPreviousEmotion().Equals("Happy") )
        {
            action1();
        }

        if( getAverageArousal() == 325.0f )
        {
            action2();
        }

        if( getHighestValence() == 0.235f )
        {
            action3();
        }

        if( getCurrentValence() > 48.0f )
        {
            action3();
        }

        if( getCurrentEvent().Equals("Text Here") )
        {
            action5();
        }

        if( !getPreviousEvent().Equals("Text here") )
        {
            action5();
        }

        if( getAverageArousal() >= 0.4f && getAverageValence() >= 0.4f )
        {
            action7();
        }
    }
```

Addendum D: Biometrics Logger excerpt

```
    private void action1()
    {
        //Write the desired code for this action.
        throw new NotImplementedException();
    }

    private void action2()    {
        //Write the desired code for this action.
        throw new NotImplementedException();
    }


    private void action3()
    {
        //Write the desired code for this action.
        throw new NotImplementedException();
    }

    private void action5()
    {
        //Write the desired code for this action.
        throw new NotImplementedException();
    }

    private void action7()
    {
        //Write the desired code for this action.
        throw new NotImplementedException();
    }
}
```

Addendum E

GameElementsModifier script

```
using UnityEngine;
using System.Collections;

using CompleteProject;

public class GameElementsModifier : MonoBehaviour
{
    //Player performance (attack and movement)
    public GameObject gameObjectPlayer;
    private PlayerMovement movement;

    public GameObject gameObjectShooting;
    private PlayerShooting shotScript;

    //Music
    public GameObject backgroundMusicGameObject;
    private AudioSource backgroundMusicPlayer;

    public AudioClip backgroundMusicHigherValence;
    public AudioClip backgroundMusicLowerValence;

    //Lightning
    public GameObject normalSceneLightning;
    public GameObject darkerSceneLightning;

    void Awake()
    {
        backgroundMusicPlayer =
                backgroundMusicGameObject.GetComponent<AudioSource>();

         shotScript =
                gameObjectShooting.GetComponent<PlayerShooting>();

        movement = gameObjectPlayer.GetComponent<PlayerMovement>();
    }

    public void alterPlayersPerformanceLow()
    {
        movement.speed = 2;
    }
```

GameElementsModifier script

```
public void alterPlayersPerformanceNormal()
{
    movement.speed = 6;
}

public void alterPlayersPerformanceHigh()
{
    movement.speed = 10;
}

public void makeEnemiesStronger()
{
    EnemyAttack.extraDamageArousal = 10;
}

public void makeEnemiesNormal()
{
    EnemyAttack.extraDamageArousal = 0;
}

public void makeEnemiesWeak()
{
    EnemyAttack.extraDamageArousal = -5;
}

public void changeAestheticsPositiveEmotion()
{
    if (backgroundMusicPlayer.clip
            != backgroundMusicHigherValence)
    {
        backgroundMusicPlayer.Stop();
        backgroundMusicPlayer.clip = backgroundMusicHigherValence;
        backgroundMusicPlayer.Play();

        normalSceneLightning.SetActive(true);
        darkerSceneLightning.SetActive(false);
    }
}

public void changeAestheticsNegaticeEmotion()
{
    if (backgroundMusicPlayer.clip != backgroundMusicLowerValence)
    {
        backgroundMusicPlayer.Stop();
        backgroundMusicPlayer.clip = backgroundMusicLowerValence;
        backgroundMusicPlayer.Play();

        normalSceneLightning.SetActive(false);
        darkerSceneLightning.SetActive(true);
    }
}

public void shootThroughBiosignal()
{
    shotScript.Shoot();
}

public bool setHappySoundtrack = false;
public bool setHorrorSoundtrack = false;
public bool lowPerformance = false;
public bool normalPerformance = false;
public bool highPerformance = false;

public bool enemiesStronger = false;
public bool enemiesNormal = false;
public bool enemiesWeak = false;
```

```
    public bool performShoot;

    void Update()
    {
        if (setHappySoundtrack)
        {
            setHappySoundtrack = false;
            changeAestheticsPositiveEmotion();
        }

        if (setHorrorSoundtrack)
        {
            setHorrorSoundtrack = false;
            changeAestheticsNegaticeEmotion();
        }

        if (lowPerformance)
        {
            lowPerformance = false;
            alterPlayersPerformanceLow();
        }

        if (normalPerformance)
        {
            normalPerformance = false;
            alterPlayersPerformanceNormal();
        }

        if (highPerformance)
        {
            highPerformance = false;
            alterPlayersPerformanceHigh();
        }

        if (enemiesWeak)
        {
            enemiesWeak = false;
            makeEnemiesWeak();
        }

        if (enemiesNormal)
        {
            enemiesNormal = false;
            makeEnemiesNormal();
        }

        if (enemiesStronger)
        {
            enemiesStronger = false;
            makeEnemiesStronger();
        }

        if (performShoot)
        {
            performShoot = false;
            shootThroughBiosignal();
        }

    }
}
```

For more information on this piece of code, please see section **5.8**.

Addendum F

GamePrototype script

The following script represents the adaption of script produced by the Rules System to match the game mechanics developed in our game prototype.[275]

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

using Biometrics;

class GamePrototype : BiometricsRulesetBase
{
    public GameElementsModifier scriptFunctions;

    protected override void awakeExtraActions()
    {
        scriptFunctions = GetComponent<GameElementsModifier>();
    }

    protected override void checkConditionsAndTriggerActions()
    {
        if( getCurrentValence() < -0.4f )
        {
            avatarSpeedLow();
        }

        if( getCurrentValence() >= -0.4f &&
               getCurrentValence() <= 0.4f )
        {
            avatarSpeedNormal();
        }

        if( getCurrentValence() > 0.4f )
        {
            avatarSpeedHigh();
```

[275] For more information, please see section **5.9**.

```
        }

        if( getCurrentArousal() >= 0.8f )
        {
            strongEnemies();
        }

        if( getCurrentArousal() >= 0.0f &&
               getCurrentArousal() < 0.8f )
        {
            normalEnemies();
        }

        if( getCurrentArousal() < 0.0f )
        {
            weakEnemies();
        }

        if( getCurrentEmotion().Equals("Sadness") )
        {
            setLevelMoodToDark();
        }

        if( getCurrentEmotion().Equals("Calm") )
        {
            setLevelMoodToDark();
        }

        if( getCurrentEmotion().Equals("Happiness") )
        {
            setLevelMoodToFunny();
        }

        if( getCurrentEmotion().Equals("Rage") )
        {
            setLevelMoodToDark();
        }

        if( getSensorCurrentValue("ShootingSensor") >= 750.0f )
        {
            shootEnemies();
        }
    }

    private void avatarSpeedLow()
    {
        scriptFunctions.alterPlayersPerformanceLow();
    }

    private void avatarSpeedNormal()
    {
        scriptFunctions.alterPlayersPerformanceNormal();
    }

    private void avatarSpeedHigh()
    {
        scriptFunctions.alterPlayersPerformanceHigh();
    }

    private void strongEnemies()
    {
        scriptFunctions.makeEnemiesStronger();
    }

    private void normalEnemies()
    {
```

```
        scriptFunctions.makeEnemiesNormal();
    }

    private void weakEnemies()
    {
        scriptFunctions.makeEnemiesWeak();
    }

    private void setLevelMoodToDark()
    {
        scriptFunctions.changeAestheticsNegaticeEmotion();
    }

    private void setLevelMoodToFunny()
    {
        scriptFunctions.changeAestheticsPositiveEmotion();
    }

    private void shootEnemies()
    {
        scriptFunctions.shootThroughBiosignal();
    }
}
```